Design • Build • Program • Learn • Compete

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

October 2011

Add A PICAXE To Your Beginner Bot

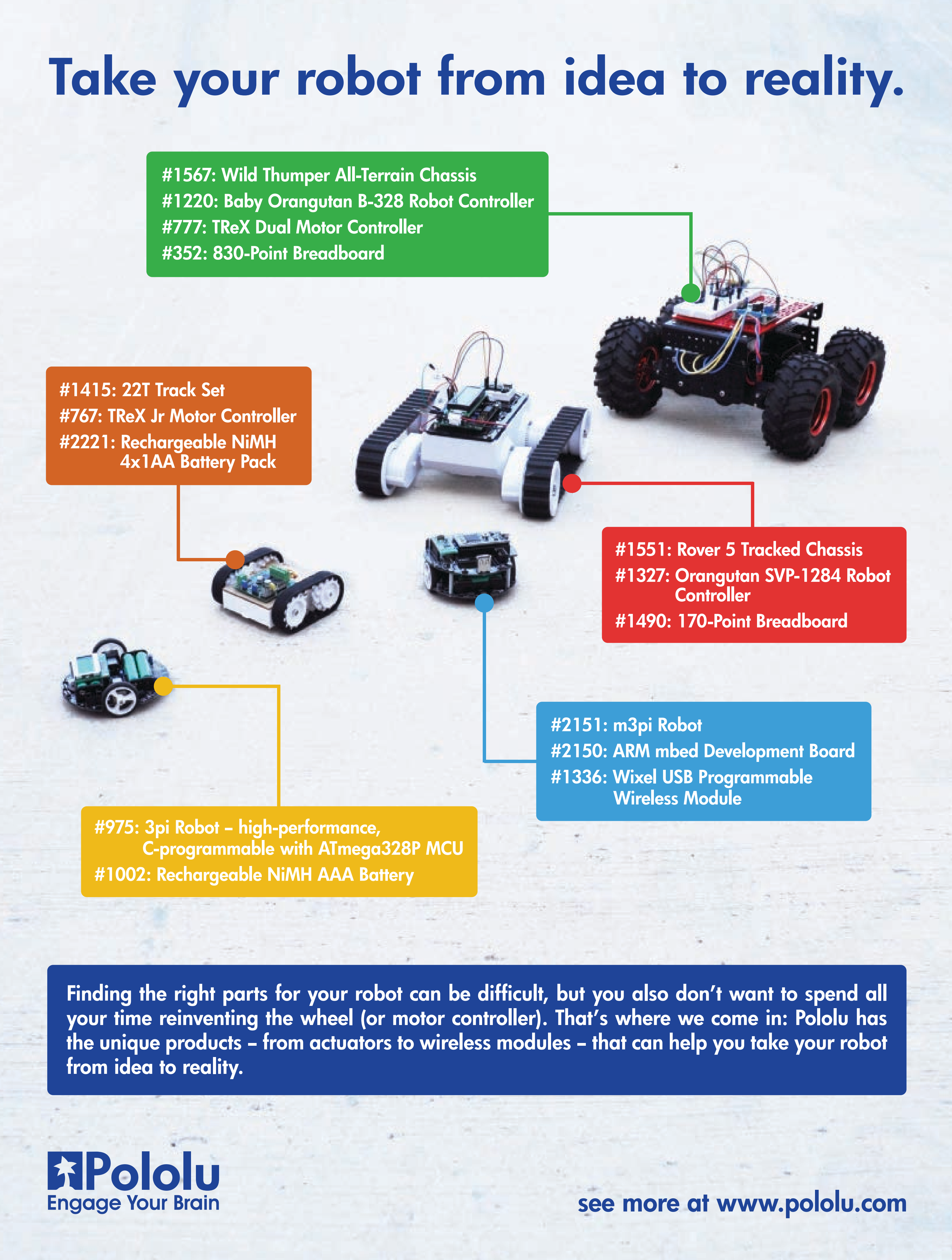
Give your robot "eyes" so it can follow the light.

- Kilobot Swarm Robots
- Extended Combat Zone
- Designing A Laser Range Finder

U.S. $5.50 CANADA $7.00

10>

0 71486 02422 4



Take your robot from idea to reality.
#1567: Wild Thumper All-Terrain Chassis
#1220: Baby Orangutan B-328 Robot Controller
#777: TReX Dual Motor Controller
#352: 830-Point Breadboard
#1415: 22T Track Set
#767: TReX Jr Motor Controller
#2221: Rechargeable NiMH 4x1AA Battery Pack
#1551: Rover 5 Tracked Chassis
#1327: Orangutan SVP-1284 Robot Controller
#1490: 170-Point Breadboard
#2151: m3pi Robot
#2150: ARM mbed Development Board
#1336: Wixel USB Programmable Wireless Module
#975: 3pi Robot – high-performance, C-programmable with ATmega328P MCU
#1002: Rechargeable NiMH AAA Battery
Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products – from actuators to wireless modules – that can help you take your robot from idea to reality.
Pololu
Engage Your Brain
see more at www.pololu.com

SERVO MAGAZINE
10.2011
VOL. 9 NO. 10

The Combat Zone...

Features

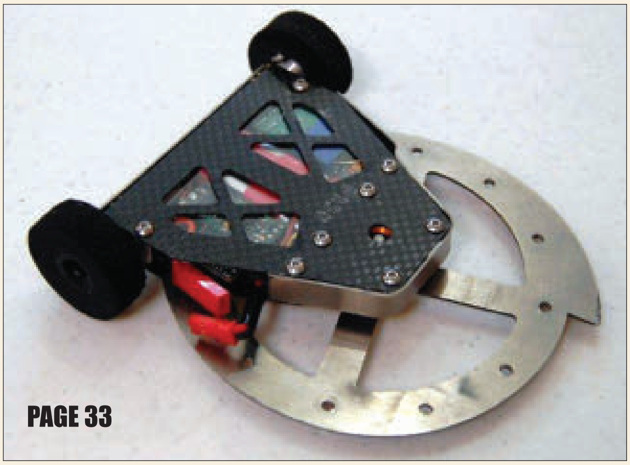
26 BUILD REPORT:
Trilobite — a Tough Beetleweight Brick/Wedge

29 BUILD REPORT:
A Team Building Exercise

36 PARTS IS PARTS:
Fingertech 'Silver Spark' Gearmotor Review

38 PARTS IS PARTS:
Susie's Saga Continues — The Beginner's Guide to Motors

40 End of the Alkalines



PAGE 33

Events

31 Upcoming and Completed Events

32 EVENT REPORT:
Clash of the Bots 2

33 EVENT REPORT:
BOTBLAST Rocks Columbia Mall

PAGE 14

Columns

08 Robytes
by Jeff Eckert
Stimulating Robot Tidbits

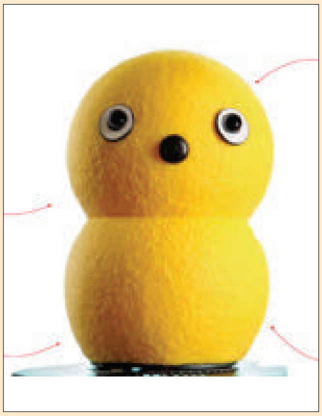
10 GeerHead
by David Geer
Kilobot Swarm Robots

14 Ask Mr. Roboto
by Dennis Clark
Your Problems Solved Here

76 Then and Now
by Tom Carroll
How Robotics Has Changed Over the Years

Departments

06 Mind/Iron
18 Events Calendar
20 New Products
22 Bots in Brief
47 Showcase
67 SERVO Webstore
81 Robo-Links
81 Advertiser's Index



PAGE 22

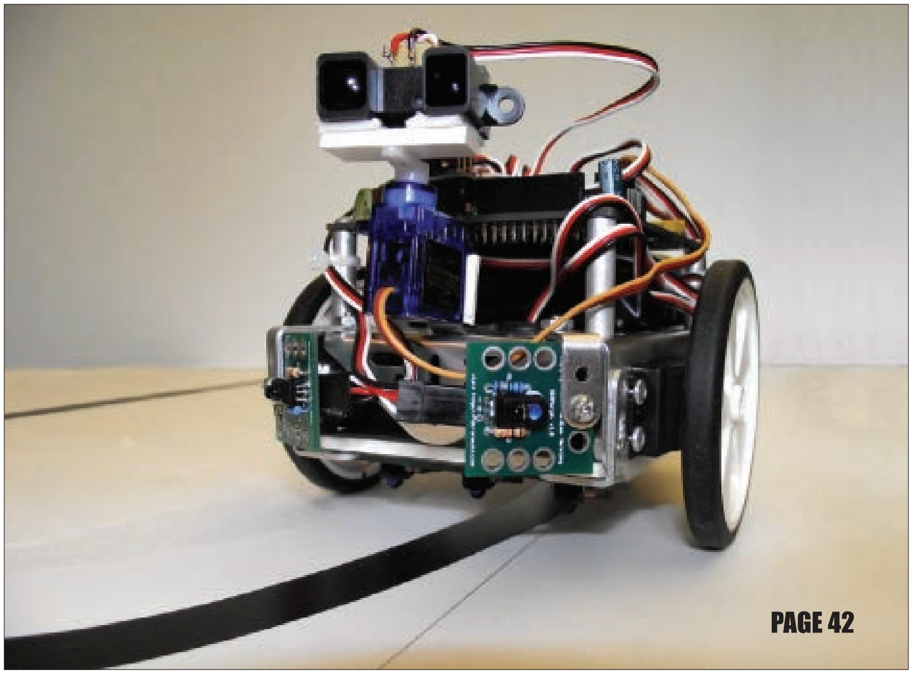
SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

42 Upgrading the Boe-Bot — Part 2

by William Henning

This last installment will show you how to add infrared remote control, line following sensors, a compass, and an infrared range sensor.

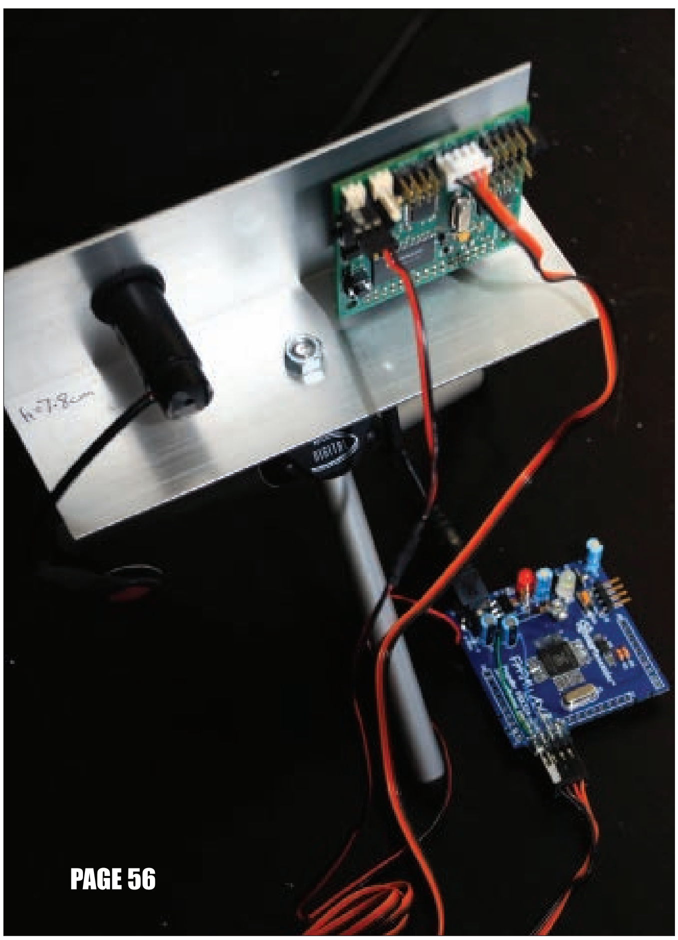


PAGE 42

48 Double Your USB Pleasure With Cerebot

by Fred Eady

What do you get when you throw a little USB, some SPI, and a PmodRF2 into a MiWi stew? If you cook up all of those ingredients in a Cerebot 32MX7 pot, you will end up with a mechanically inclined 32-bit RF-capable robotic controller. This month, Fred shows you how to use various Digilent Pmods and the Microchip MiWi stack to establish an RF data link between a pair of Digilent Cerebot 32MX7s.



PAGE 56

56 Designing A Low Cost Laser Range Finder — Part 1

by Joe Grand

Joe takes us on his journey in developing a laser range finder, from beginning concepts to finished product.

62 Getting Started With FPGAs — Part 2

by David Ward

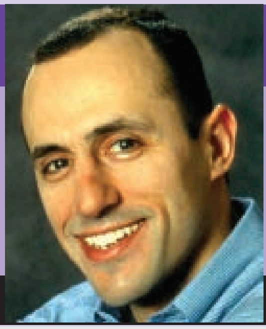
For this last part, a more complex and useful digital design using the Basys2 FPGA trainer will be demonstrated.

70 Adding a Microcontroller to the Beginner Bot — Part 3

by Gordon McComb

By moving up to a miniature computer to operate your Beginner Bot, you'll be able to modify its action and behavior just by rewriting a few lines of code.

Mind / Iron



by Bryan Bergeron, Editor

The Long Arm In Space

This summer — with the launch of the Atlantis — marked the end of the NASA Space Shuttle Program and, along with it, the use of that amazing robotic arm. The Canadarm — first flown on a shuttle mission in 1981 — has six joints, is 50 feet long, and can lift 586,000 pounds in space. I'm not sure how to interpret that last figure, given things are supposedly weightless in orbit, but it's fascinating to note that the hollow arm can't even support its own weight on the ground.

In the '80s, a major breakthrough in robotics was the ability to control the arm with a joystick. Today, the joystick as a human interface to robotics is commonplace. The fly-by-wire joystick and associated algorithms are prominent additions to the growing list of space spinoffs. As a roboticist, it's easy to get caught up in the strength of materials, mechanics of the two shoulder joints, and other technical issues. However, the greatest contribution of the Canadarm to robotics is that it has provided inspiration for thousands of would-be and soon-to-be scientists, engineers, and astronauts.

The inspiration of something just at the edge of what's possible can change the trajectory of open minds of any age. Who wouldn't want to operate the Canadarm in space, or to be involved in creating a new and improved version for, say, a Mars mission? With the privatization of space transport, it's possible that there's a spot for you or someone you know to work on the next robotic arm or other robotic program that will directly impact space exploration.

I can still remember as an engineering student, visiting the NASA Michoud Assembly plant just outside of New Orleans, where they produced external fuel tanks. I remember the stacks of aluminum rings that formed the internal skeleton of the tanks, lathed to a few thousandths of an inch. Then, there was the huge robot finisher that rotated a fuel tank while spraying on a thin coating of sealant. It wasn't my first exposure to robotics, but it was the most significant. I can still remember every machine and robot in the plant, and thinking how fortunate the swarms of engineers and scientists were to be part of the effort.

What's the next big inspiration for robotics engineers and enthusiasts? I can't point to a singular project as prominent and obviously critical as the Canadarm. However, there are niche projects that each draw significant followings. The military is obviously heavily invested in robotics — from studies on autonomous supply trucks to rescue robots that can extract a wounded soldier from the field.

Of course, there's the entire weapons development field, with smart missiles that can navigate and track their targets autonomously. It'll probably be some time before developments in military robotics trickle down to consumer goods.

In the meantime, there is the growing field of surgical robotics which promises better results with smaller incisions and shorter healing times. There's the home robotics market that promises to make our lives easier as we become old, arthritic, and less mobile. In short, there's more than enough 'important' work to do in robotics. It's simply a matter of enthusiasm, focus, and preparation.

By the way, the retirement of the shuttle fleet doesn't mark the end of robotic arms in space. The Space Station's Canadarm2 (also made in Canada) remains a useful tool to the astronauts. For more information on the Canadarm see **www.nasa.gov**. Check out **www.sti.nasa.gov/tto** for details on commercial spinoffs from the space program. SV

FOR THE ROBOT INNOVATOR
SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Kevin Berry, Gordon McComb, David Ward, William Henning, Joe Grand, Fred Eady, Dave Graham, Pete Smith, Morgan Berry, Thomas Kenney

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

You Asked and We Listened — New Forums are ONLINE!

This issue marks the first time that *SERVO Magazine* content will be directly linked to our online presence! We have completely rearchitected the forums so they now closely reflect the content of the magazine. Every regular column now has its own dedicated forum with the column author as your host! In addition, there are also forums to support project articles for reader projects and even an area where you can learn about what it takes to become a writer for the magazine!

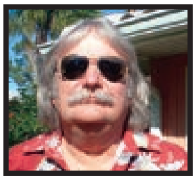
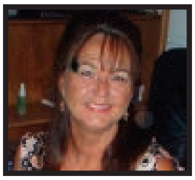
This new forum design invites you to interact with the author(s), read about corrections or modifications to projects, post questions, or just find and interface with lots of other folks who enjoy the same things you do.

As an added bonus, we have populated the forums with example articles by each author so you can read them there if you don't have time to read them here.

We hope you find the new forum layout exciting and useful. Please come by, have a look around, and introduce yourself! To explore the new forums, just point your browser to: **http://forum.servomagazine.com**.

Hope to see you online soon! :)

Vern Graner, Forum Moderator, *SERVO Magazine*

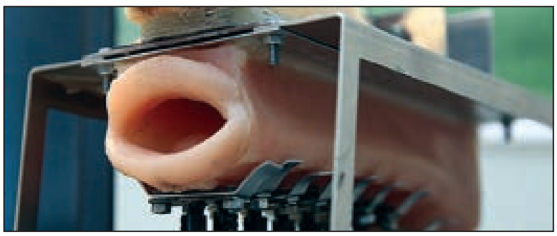
Robytes

by Jeff and Jenn Eckert

Mechanical Mule Off To Afghanistan

Lockheed Martin's SMSS will be going to Afghanistan for evaluation.

Back in August, we reported that Army research scientists had expressed disappointment in the performance of autonomous vehicles deployed in Afghanistan and were actually contemplating the use of pack mules instead. Apparently, the Army Rapid Equipping Force isn't quite as pessimistic, as they will be sending four Lockheed Martin (**www.lockheedmartin.com**) Squad Mission Support System (SMSS) vehicles for a three-month field evaluation next month. The 11 ft SMSS — said to be the largest autonomous vehicle ever deployed with infantry — can carry more than 1,000 lb (450 kg) of a squad's equipment over rugged terrain — including through shallow water — with a range of 125 miles. It features a choice of three operating modes: supervised autonomy, tele-operation, and manual control. A sensor suite mounted on the front allows it to lock onto a 3D image of a particular person and follow along, or it can navigate along a series of GPS waypoints. A fifth unit will remain in the US for further analysis. At present, the SMSS is basically just an unarmed mechanical beast of burden, but the Army says, "The long-term vision of this system can accommodate armed variants, while improving its reconnaissance, surveillance, and target acquisition capabilities within the concept of supervised autonomy."

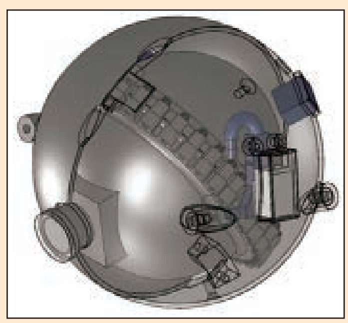


Strange robot mouth developed at Kagawa University.

Straight from the Silicone Mouth

A bot doesn't have to be animal inspired to be creepy and disgusting, as evidenced by a device concocted at Japan's Kagawa University (**www.kms.ac.jp**). At the last Robotech Expo in Tokyo, Prof. Hideyuki Sawada demonstrated a mechanical mouth that is about as near to the real thing as you would ever want to see. It has lips, eight vocal chords, a tongue, a nasal cavity that provides resonance, and a mechanical pump that supplies air to the vocal chords. The shape of the mouth is manipulated by a set of actuators, allowing it to form words. It can even listen to itself via a microphone and make adjustments to sound more "human." It seems a little odd to put together something this elaborate when a cheap speech synthesis device would accomplish pretty much the same thing, but it is more visually entertaining this way. You can check out the vid by aiming your browser at **www.jkeckert.com/freedownloads/robotmouth.mp4**. It sounds even weirder than it looks.

Cannonball Bot to Inspect Nukes



MIT's cannonball bot, intended for nuke plant inspections.

One of the side effects of the Fukushima power plant disaster has been increased concern about other nuke plants around the world, particularly those of venerable vintage. This summer, the Associated Press said that a study of documents obtained from the US Nuclear Regulatory Commission revealed that tritium (a radioactive form of hydrogen) has leaked from at least 48 or 65 commercial sites from corroded, buried pipes that transport water to cool the reactor vessels. Many have gone uninspected for 30 to 40 years. The good news is that none of the leaks has reached public water supplies, but it seems like a pretty good idea to start inspecting the piping. As it turns out, an AUV system for doing just that is being developed at MIT's d'Arbeloff Laboratory (**darbelofflab.mit.edu**) by Prof. Harry Asada. The tough part was designing an inspector bot that can't get trapped in the reactor structures which eliminates standard designs that use propellers or rudders. Hence, Asada's design uses a propulsion system that employs the force of water rushing through the reactor. The unit is controlled via a network of Y-shaped valves embedded in the bot's skin that direct the water's flow toward little windows that create a propulsion jet stream and propel the robot in the opposite direction. An onboard camera takes photos of the pipe’s interior and uses a wireless system to transmit the images in real time. When fully developed, the units are expected to be cheap enough to use as short-term disposable patrollers, not only for nuclear facilities but for municipal sewer pipes and other difficult to inspect installations.

www.servomagazine.com/index.php?/magazine/article/october2011_Robytes

Worm-Bot Slithers Around Obstacles

Scanning electron micrograph of a nematode. Courtesy of the Max Planck Institute.

This month's entry to the category of "bots inspired by yucky animals" comes from Jordan Boyle at the University of Leeds (**www.leeds.ac.uk**). His "worm-bot" is patterned after the nematode Caenorhabditis elegans — a 1 mm roundworm that slithers around in temperate soils. C. elegans has the ability to move within a wide range of speeds by varying its wiggle frequency, and yet it has almost no detectable neural center. According to Boyle, "It has an unusually small nervous system, comprising just over 300 neurons. Rather than using a central neural subcircuit as a pattern generator, it seems to generate its undulatory motion using around 100 neurons in a way largely driven by feedback from stretch sensors along its body." Much like its inspiration, the worm-bot doesn't pay any attention to its surroundings; it just keeps on wiggling until it gets where it's going. The machine differs from C. elegans in various ways, particularly in being about 2 m (6.6 ft) in length and having a rigid, snake-like backbone. Given an appropriate skin, the next model should be able to swim in water or crawl through mud. The final version is envisioned as a useful tool for search-and-rescue operations in which it look for survivors trapped in collapsed buildings. **SV**

GEERHEAD

by David Geer

Contact the author at geercom@windstream.net

Kilobot Swarm Robots

A Swarm You Can Program, Power Up, and Charge Collectively

The idea behind swarms of robots is that they can accomplish more together than they could individually. This only works if the number of robots that can be programmed and managed in a swarm collectively (rather than individually) scales up easily to hundreds and thousands of robots that can accomplish massive tasks.

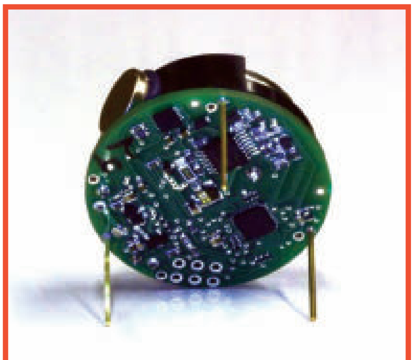
Until now, swarms have not been up to the task due to the cost of multiple robots, assembly time, and the simple lack of a way to program them, start them up, and recharge them as a group (rather than one by one). Kilobots bring us a step closer with affordable robots ($14 each) that can be assembled in five minutes each and which can scale up (for now) to a collective of 25 robots working together.

Design

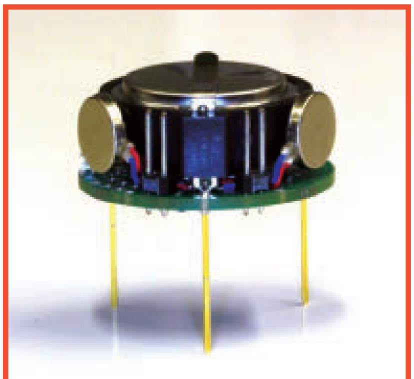
The roboticists in the Computer Science Group at Harvard University designed Kilobots so each could move in an environment, run a "user defined" program, communicate with neighboring Kilobots, measure the distance to its neighbors, display something about its internal state to assist with debugging, and allow for scalable operations, according to the paper, "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors" by Michael Rubenstein, Nicholas Hoff, and Radhika Nagpal.

To enable these capabilities, the designers endowed the robots with vibration motors, lithium-ion batteries, rigid supporting legs, infrared transmitter/receivers, and three-color RGB LEDs. The robot test environment included 25 Kilobots, a laptop-based control station, an overhead controller unit, and a charging station on a flat, smooth, level reflective table.

The simple means of movement for these robots is not wheeled motion — as popular as that design option may be — because it was simply too expensive. Instead, the robots each have two "sealed coin-shaped vibration motors."



Bottom of a Kilobot with three legs and circuitry.



Standing view of a Kilobot.

When one of these motors is activated, the centripetal forces generated by the vibrating motor are converted to a forward force on the Kilobot located at the motor's mounting location. The vibration of one motor [causes] a rotation of the Kilobot about its vertical axis in one direction, while the other motor rotates the robot in the other direction. This is an application of differential drive where the robot can move forward, and in clockwise and counterclockwise rotations, according to the Kilobot paper.

While this method provides no "odometry" and no measure of distance over time can be made for individual robots, the collective can

use the measured distances between the neighboring robots as feedback to correct errors in the robot's movement. This enables reasonably accurate motion control of the individual robots and the collective.

Rough terrain is also out of the question for this brave little band of bots. This is another limitation that is necessary to keep the cost of the collective down, but it does not impede the type of behaviors the researchers wish to observe.

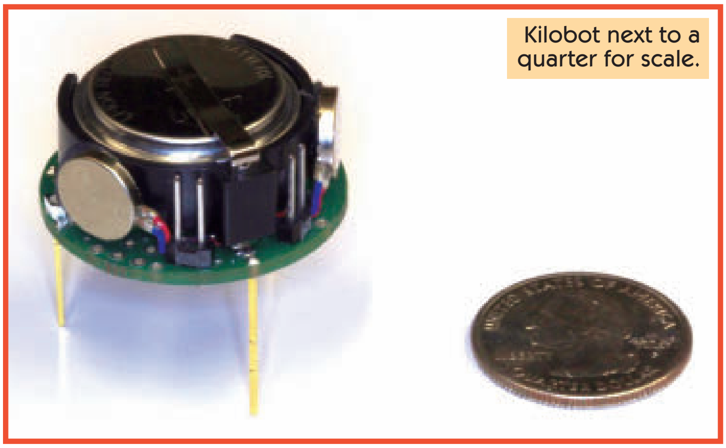
The Harvard roboticists included communications between robots about their distances from each other because any swarm robot work with merit offers communications and sensing between robots. The Kilobots use infrared LED transmitters and photodiode receivers located in the middle of the PCB and aimed directly downward at the table to accomplish these communications. "Both the transmitter and receiver have an isotropic emission or reception pattern which allows the robot to receive messages equally from all directions," per the Kilobot paper first referenced above. Any nearby robot can receive the transmitted light which bounces off the table at an angle and upward to surrounding robots.

Because the robots all communicate on the same channel, there is a risk of collisions. To avoid this possibility, the robots use the standard Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique. Nearby robots can use up the infrared bandwidth through collisions, but the channel was sufficient to support the 25 robots.

The recipient of the communication measures the incoming signal strength (infrared light intensity) to determine the distance between the receiver and transmitter of the communication.

The robot's controller communicates with the robot's low level electronics and runs user-defined behavior programs. The controller hardware is an Atmega328 microprocessor running at 8 MHz with 32K of memory. The controller uses two pulse width modulation (PWM) channels to control the motor speed. It uses 10-bit analog-to-digital converters to measure infrared light intensity. Its self-programmable memory updates the robot's program. Finally, the controller comes with a low power sleep mode. The robot's programming is written in C for expedient behavior programming.

The robot's power emanates from a 3.4 volt 160 mAh lithium-ion battery. The battery powers the robot for three to 10 hours, depending on the robot's activity frequency. The battery attaches to three voltage regulators and a battery charger. Two regulators attach to the motors and the communications system. Because the microcontroller can turn the regulators on and off, both the power and communications can be powered on and off to conserve power. The third voltage regular affords the microcontroller continuous operating power. The charger kicks in whenever it receives 6 VDC and stops when the battery is charged.



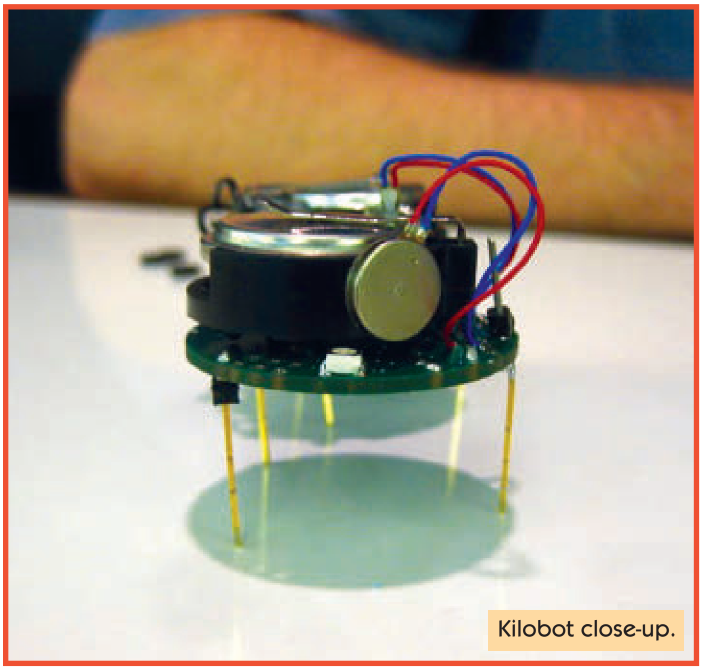
Kilobot next to a quarter for scale.

At $14 each, the Kilobots are ten times less expensive than the next least expensive robot in any known robot swarm. The robot's total cost can be itemized by cost of locomotion, power, communications and sensing, control, structure, and miscellaneous. Unless the robot can be quickly assembled, the man hours involved can create an unreasonable added cost. Most of the Kilobot's parts are surface-mount and can be added via a pick-and-place manufacturing robot.

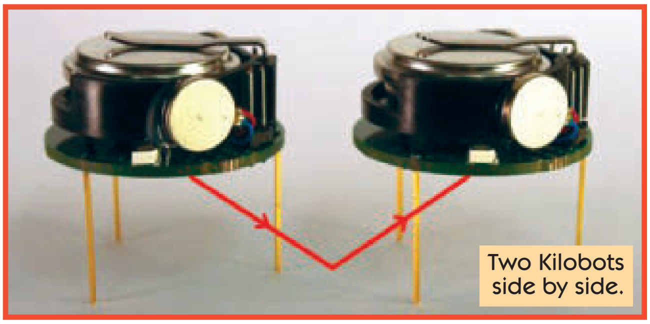
The rest of the parts including the legs, battery holder, motors, and infrared components are assembled by hand or by using custom-made assembly rigs. Total assembly time from start to finish is under five minutes per robot, or 125 minutes for the 25 in the collective.

Collective Capabilities

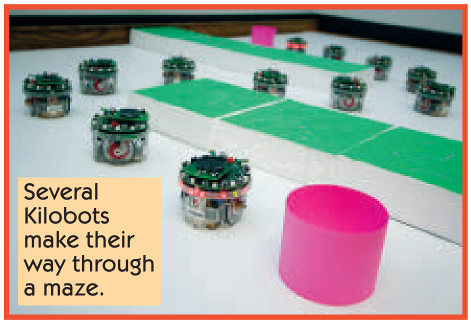
What one Kilobot cannot do, 25 can do with ease. The robots demonstrate the ability to move around quite a bit



Kilobot close-up.



Two Kilobots side by side.



Several Kilobots make their way through a maze.

while remaining in their environment, communicate to and measure the distance to their robot neighbors, and to run a controller unit. The first activity to demonstrate these functions is one in which the robot orbits or traces a circle around another robot which then functions as the center of the orbit or circle.

The stationary robot sends a message at one-tenth of a second intervals and the orbiting robot receives them to maintain constant awareness of where the center of the orbit or circle is. Using these distance communications, the orbiting robot's controller calculates when the robot deviates from the orbit so that — using a PD controller — it can adjust motor intensity to correct its course.

In another demo, a Kilobot followed a complicated path; a U shaped path to be exact. To set up this demonstration, three Kilobots form a triangle and know their own positions. They communicate that position to the moving Kilobot 10 times per second. The moving robot uses those positions and its distance to those positions to determine its position in the U shape that it must follow. The robot must move from inside the first robot to outside the second one, and back inside the third one to complete the U shaped path.

The Kilobot collective can power itself and its members on and off, plug itself in to recharge or receive programming, and start, pause, or stop the program without any human intervention. For one human operator to command the collective as a whole, the robots have a single overhead infrared controller that transmits infrared messages to the collective. That controller is commanded by the operator from a computer.

To power the robot on and off, it has a sleep mode for off instead of a battery disconnection, and it can power on again after one minute. The microprocessor then wakes up and turns on its infrared communications. The overhead controller sends a wakeup message every 3 ms to tell the robot to wake up the rest of the way. The overhead controller can also switch the robot back to sleep mode. The robots can remain in sleep mode for up to three months on a single battery charge. The whole collective can be turned on in under a minute using the overhead controller.

The roboticists charge the collective by pushing them all simultaneously onto a conductive surface with a stick. A conducting board is placed on top of the robots. The appropriate voltage is applied to the bottom conductive surface and the top conducting board which connects the input of all the robot's chargers to the given voltage. The whole process can be completed while addressing the robots as a whole rather than one by one.

The robots are collectively programmed by transmitting a jump to a bootloader message from the overhead controller which — when received by the main program code — causes the program counter to move to the bootloader section of the code, executing the bootloader program. This way, all the robots in the collective are programmed in under a minute.

Aerial view of the maze.

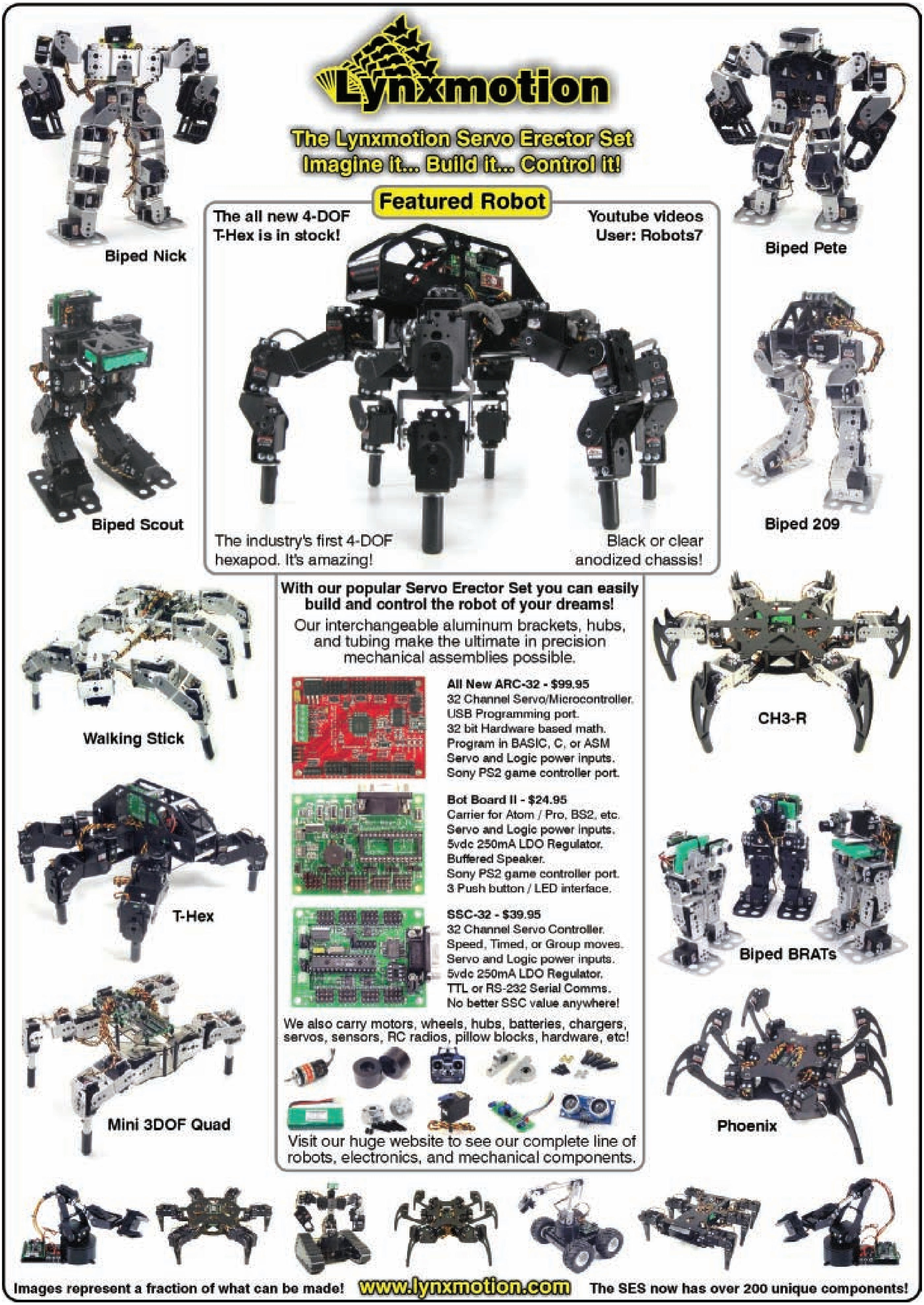
Conclusion

As small as the Kilobot collective is, it accomplishes much in taking swarm robotics to the next phase: scalability. **SV**

Resources

Michael Rubenstein, Kilobots creator
http://people.seas.harvard.edu/~mrubenst

More images of Kilobots
https://picasaweb.google.com/radpicasa/swarms


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
The all new 4-DOF T-Hex is in stock!
Youtube videos User: Robots7
The industry's first 4-DOF hexapod. It's amazing!
Black or clear anodized chassis!
Biped Nick
Biped Pete
Biped Scout
Biped 209
Walking Stick
CH3-R
T-Hex
Biped BRATs
Mini 3DOF Quad
Phoenix
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.
All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Visit our huge website to see our complete line of robots, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
The SES now has over 200 unique components!

Our resident expert on all things robotic is merely an email away.

roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

by
Dennis Clark

ASK MR. ROBOTO

I'm going to kill a few birds with one stone this month. I've been quizzed on converting a toy into a robot (always fun, and I don't have to design the mechanical!), PWM creation for cheap motors, and basic robot "vision" sensors. I'm going to answer all these questions at once with a little project that I did in two phases, a couple of years apart. I'm going to show you how I converted a toy IR remote control spider into a robot that won't walk off the edge of a table. This is a fun one for the kids.

Speaking of kids, this robot was built with the intent of making it as robust as the toy was, with all electronics and batteries, wires, etc., fully enclosed. Nothing to break! My little spider has withstood the test of time with Pre-K kids, Kindergarten, and all the way to fourth grade students, without a single failure. I'd call that a success! Let's begin!

Figure 1 shows the spider as it came in the box from **WowWee** toys. These folks come out with the coolest platforms to hack into robots. They are the company that gave us RoboSapien, RoboReptile, and many others. I applaud their gifts to the robot hobbiest.

Quite by accident, I found a spider that didn't have the silver paint on it, it was all red – perfect! The first thing I did was completely strip out the electronics that it had in it. I left the array of LEDs on the bottom (which I've not found a use for yet), the motors (two of them), and the battery pack which was molded in anyway.

This is a fascinating platform. There are two motors, each of which runs a side of legs. The thing walks by alternately lifting pairs of legs on each side. When the motors are in sync, it walks straight. We all know that without feedback, no two motors EVER run at the same speed, so the spider often kind of "crabs" to one side. It does mostly run well, and quietly, and kind of creepy. Perfect for kids and robots.

Figure 1. WowWee spider.

After you remove the original "guts" of the spider, you are left with precious little space, so think carefully. I designed this controller with through-hole components on a simple prototyping board. I could do a *lot* better with a specially designed PCB or even surface-mount devices, but this was a "one off" project for fun, so I broke out the soldering iron after I figured out what I wanted.

I started with a simple microcontroller design running a 754410 dual H-bridge chip at 1 kHz PWM rate. I then added one of my custom IRPD boards, carved and shrunk down. For the

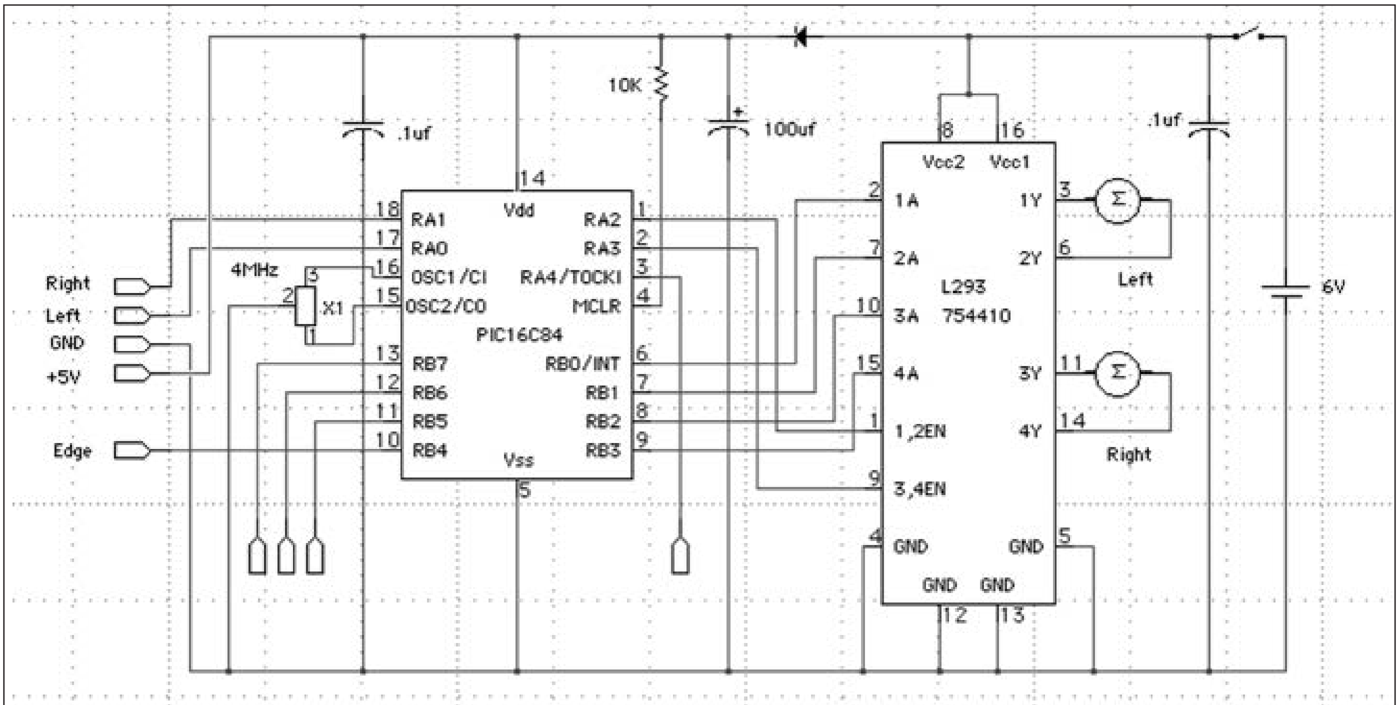


Figure 2. Spider controller.

schematic and assembly source for this device, check out my webpage at **http://techtoystoday.com/projects/botproj.htm** or in the downloads at the article link. Look for the **12C508 IR Obstruction Detector** project. You'll find a description, theory of operation, and source code, all right there and free. This controller and sensor board gave me a simple "bump-n-go" walking spider robot. See **Figure 2** and **Figure 3** for the controller and IRPD schematics.

This was fun, but just not enough. I should say that I also started out by using some ancient PIC16F84 parts. You should go for some newer parts like the 16F628 or 16F660, or newer stuff. (The 16F84 is *so* 20th century!) My later design used a PIC16F628 that I found in my junk box.

You will note that I don't have a voltage regulator in my design. Yup, the PICs will operate at most voltages from 4V to 5.5V, and with four AAA batteries my max voltage was 6V minus a .6V diode drop which is 5.4V. Yeah, sloppy engineering, but this was a design on a shoestring! If I was to do it again, I would design with 3.3V parts and a low dropout voltage regulator for more reliability. However, so far, it has worked.

Anyway, I wanted the robot to stay on a table most of the time, or to stay on my "inverted" mini Sumo board which is white in the middle and has black edges. This meant that I needed something that would reflect off a normal table while looking down to avoid going off the

Figure 3. IRPD circuit.

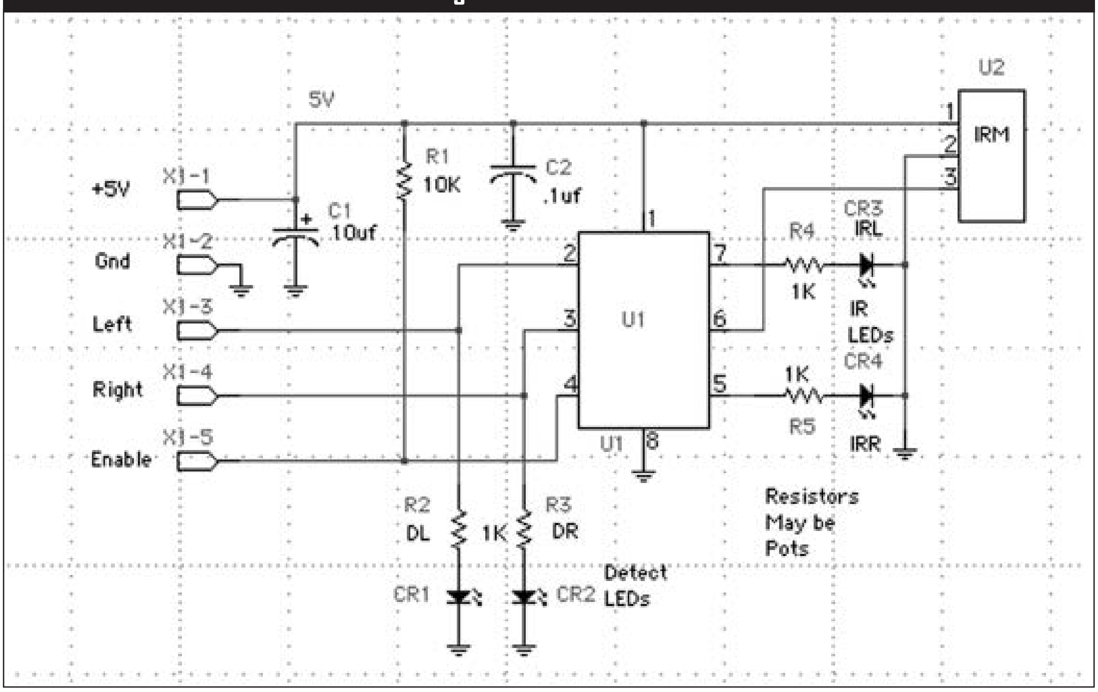


Figure 4. Looking down on the Sharp IR sensor.

Figure 5 shows where everything went.

I used two speeds of PWM for this project: *slow* and *fast*. I think I could have gone a little faster though. That will be a modification for the next round of hacks on this guy! I used the CCS PCM compiler to write the code in C for the spider robot. Since these PIC processors don't have PWM timers, I faked it in the ISR routine by just counting. It worked fine. **Listing 1** shows how I created my PWM with a simple counter interrupt. I only had four bits of resolution, but really, if we don't use feedback, how many speeds does a robot use? In my case, typically three: off, slow, and fast.

In **Figure 5**, you can see how tight the fit of everything is. Nothing a little tape and hot glue gun can't handle, though.

I like my robots to look interesting, but this time, the look was accidental. Take a look at **Figure 6**. What a face! This robot initially scared my three year old. She did eventually get used to it, though.

Rather than print the source program for this robot, I've made it available on the *Servo Magazine* website as *bugf84.zip* in the article downloads. The full source is there and it implements only three behaviors: *wander*, *avoid,* and *edge*. I'm a "Brooksian" behavioral programmer, so if you are interested in what Rodney Brooks called subsumption programming, this robot code is a gentle introduction.

To recap, this robot is a hacked toy made fully autonomous. It will wander around and avoid colliding (mostly) with objects in its path. It will avoid walking off the edge of a table, within reason. Within what reason, you ask? The Sharp IR detector's beam will bounce right off of a highly polished surface and the robot will act like it sees the edge of the table all of the time. The same thing will happen if it is walking on a dark fluffy

cliff. I found a nice device built by Sharp, and as usual, it has a huge, complex incomprehensible part number. I used a Sharp GP2Y0D810Z0F digital distance sensor that I found at **www.pololu.com**. This model has a 10 cm detection range. Perfect for my little walker. See **Figure 4** for what it looks like.

The controller, IRPD, and Sharp sensor were pretty tough to shoe-horn into this little machine, but I did it.

Listing 1: Simple PWM interrupt routine.

```
#int_rtcc                          // This function is called every time
void clock_isr(void)
{                                  // the RTCC (timer0) overflows (255->0).
    if (left == 0)
        {
        output_bit(PIN_A2,0);
    }
    left--;
    if (right == 0)
        {
        output_bit(PIN_A3,0);
    }
    right--;
    if (PWMperiod == 0)
        {
        PWMperiod = SPERIOD;
        left = sleft;
        right = sright;
        output_bit(PIN_A2,1);
        output_bit(PIN_A3,1);
    }
    PWMperiod--;                   //C int maint makes this > 32us!
    set_rtcc(240);                 //255-239 =16, 16*2us = 32us
}
```

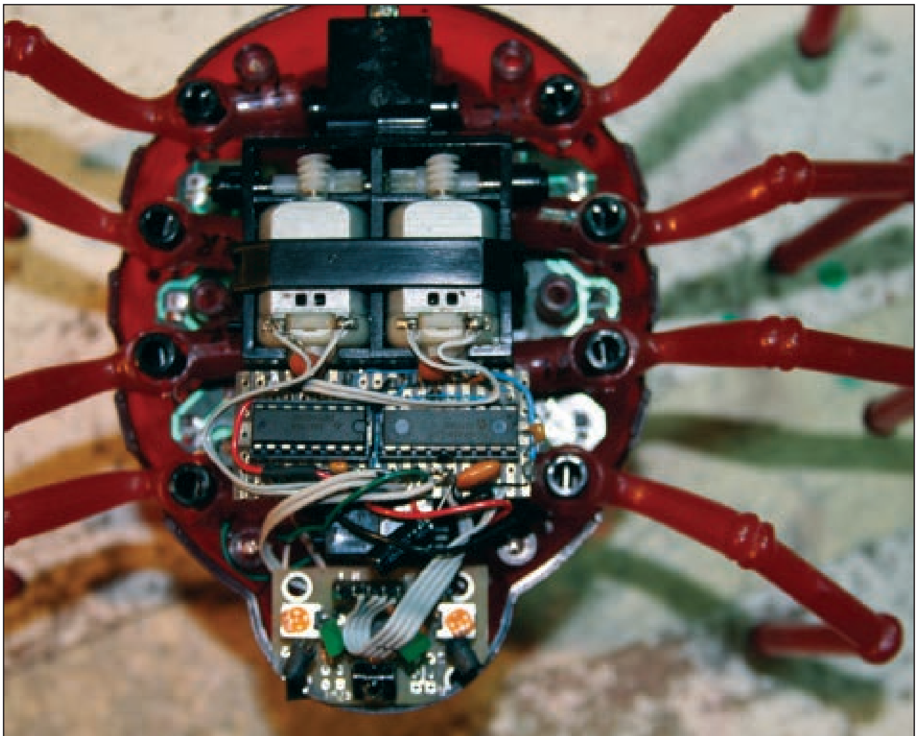


Figure 5. Spider controller electronics.

carpet. So, it isn't perfect.

I used this code example to explain subsumption programming over a year ago using state machines, but didn't go into its construction much. So for those interested, this will allow you to check out that article (December '10) with this one, which focuses on how to construct the electronics and hardware. Have fun!

As always, if you have a question for Mr. Roboto, drop me a line at roboto@servomagazine.com and I'll be happy to work on it! Until next time, keep on building those robots! **SV**

Figure 6. A face only a spider could love!

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— **R. Steven Rainwater**

OCTOBER

15 **The Franklin Cup**
Philadelphia, PA
Various weight classes of remote control vehicles attempt to destroy each other at this annual NERC event.
www.nerc.us

20-21 **Competencia Robotica (LARC)**
Center for Robotics UTFSM, Valparaiso, Chile
This year's events are Robot Freighter and Police Robot. Robot Freighter is a way-finding contest in which robots must pick up a payload and take it to a destination. Police Robot is a contest where autonomous robots must surveil a city, locate criminals, and deploy guard robots in areas with too many criminals.
http://robotica.elo.utfsm.cl/competencia

21-22 **CalGames**
Archbishop Mitty High School, San Jose, CA
This is a FIRST-based robotics event for high school teams.
www.wrrf.org

21-23 **Critter Crunch**
Hyatt Regency Tech Center, Denver, CO
Autonomous and remote controlled robots, along with other machines battle each other in hopes of winning prizes that include handmade objects by local artists and awards for "amusing and arbitrary achievements.
www.milehicon.org/?page_id=16

22-23 **Chibotica**
iHobby Expo, Donald E. Stephens Convention Center, Rosemont, IL
Lots of events to keep your robot busy including line following, maze solving, mini Sumo, and a talent show.
www.chibots.org

28-30 **COMBOTS Cup**
San Mateo, CA
Remote control vehicles destroy each other.
http://combots.net

NOVEMBER

6 **International Micro Robot Maze Contest**
Nagoya University, Japan
This competition is held along with the International Symposium on Micro Mechatronics and Human Science. Events include 1 cm Micro Robot Races, Teleoperated Mountain Climbing Micro Robots, the Autonomous Micro Robot Maze, and Micro Biped Locomotion Robots.
http://imd.eng.kagawa-u.ac.jp/maze

12 **AHRC Robot Rally**
Pinckneyville Community Center, Norcross, GA
This year's competition includes Open Competition, R/C Qube Quest, Maze Solving, and a new six part Polyathlon. There will also be mini Sumo exhibition matches.
www.botlanta.org/robot-rally

12-16 **Real World Robot Challenge**
Tsukuba Expo Center, Tsukuba, Japan
Autonomous robots compete in the real world, literally — they must navigate the Tsukuba streets and sidewalks, coexisting with humans, animals, and vehicles.
www.ntf.or.jp/challenge

20 **Robocon**
Tokyo, Japan
Ssixty-two technical colleges and 57 other schools nationwide participate in Robocon, which culminates at this championship.
www.official-robocon.com

COMBOTS CUP V



$3,500 ain't Shrapnel!

What could be better than crushing your opponent to dust in the robot combat arena? *Getting paid for it!*
Come to San Francisco this winter for the fifth annual ComBots Cup - **The biggest cash prize in combat robotics!**

Heavyweight, Middleweight, Lightweight, and Featherweight robots are all invited to go head to head with the best robot combat teams in the nation. But the biggest bots get the biggest reward - The heavyweight winner takes home the 100 pound ComBots Cup trophy and a nice juicy check! The first year, **Sewer Snake** devoured **Karkas** for the big money. In 2007, **Brutality** muscled the win out of **SewerSnake's** jaws. In 2008, it was **Last Rites** who cut down the competition and took home the prize. Last year, **Original Sin** finally grabbed the Cup. This year - it could be you!

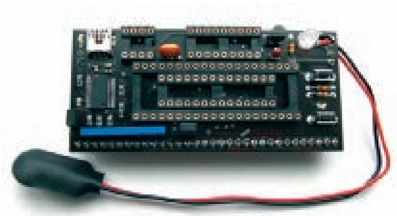
Get out your spare motors, over-volt your speed controllers, grease up your sprockets and get ready to destroy the competition! Full details at ComBots.net - ***Register now!***


October 29-30th, 2011 in San Mateo, CA
http://ComBots.net or cup@combots.net

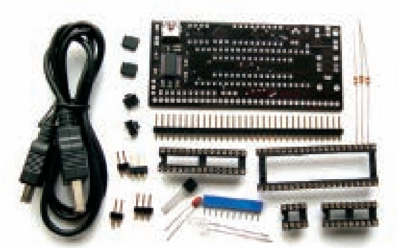
ROBOT RFL FIGHTING LEAGUE

NEW PRODUCTS

KicStand Development Board



Assembled $45



Unassembled $35

The KicStand development board, available from ISORC Technologies, is designed especially for experimenting with the KicChip brand of microprocessors and will fit on any standard breadboard.

With features like the free PBasic KicStudio programming software which also allows flowchart programming; choice of use with the eight-, 18-, 28-, or 40-pin KicChip (or PICAXE) processor; onboard USB converter; ability to select pull-up, pull-down, or floating inputs; onboard programmable LED and input button; reset button, reverse polarity and over voltage/current protection; drop-in compatibility with select PICAXE processors; rugged machine pin sockets for years of use; and optional upgrades like an external resonator for faster processing, 9V pig-tail, and components for non-breadboard use, this design is arguably one of the easiest possible ways to get started designing and programming microprocessor projects.

The KicStand comes as a semi-kit (surface-mount components are preassembled) or you can get it completely assembled. Either way, the kits come complete with the USB cable, KicStand Support CD, and all four chip sockets. The KicStand kits do not include the processor or optional components.

KicChip processors are sold separately so the end user can decide what best fits their needs.

KicChip processors prices are:

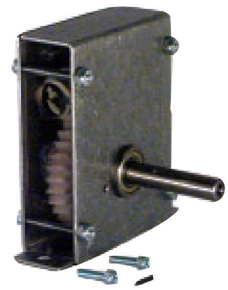
- Eight-pin — $2.95
- 18-pin — $8.45
- 28-pin — $6.95
- 40-pin — $7.95

For further information, please contact:

1SORC Technologies, LLC Website: **www.1sorc.com**

New Gearboxes

AndyMark, Inc., announces two new products: WormBox and CIM-Sim gearboxes. The appeal for both gearboxes is their lightweight design and mounting options.



Inspired by the common garage door opener, the WormBox is a quiet, lightweight gearbox designed as a right-angle drive and for utilizing the popular 2.5” CIM motor. This gearbox can be used as part of a four wheel drive system for a 150 lb robot or a lightweight robot arm. It has a reduction ratio of 16:1 and weighs only 1.16 pounds (without the motor). This gearbox comes fully assembled. More specs, photos, and a video demonstration of four WormBoxes with mecanum wheels are available at **www.andymark.com/wormbox**.

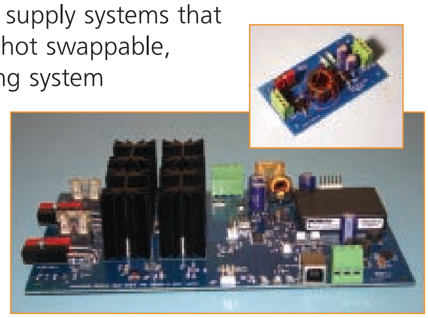
The CIM-Sim gearbox is another new lightweight gearbox which is designed to accept one or two RS-500 series motors. It has the same output shaft and mounting hole dimensions as the popular 2.5” CIM motor, and a reduction ration of 5:1. It weighs just 0.55 lbs. This gearbox ships as an unassembled kit of parts. Photos are available at **www.andymark.com/cimsim**.

For further information, please contact:

AndyMark, Inc. Website: **www.andymark.com**

Modular Power Supply Components for Robotic Systems

rChordata, LLC has announced its new PowerBotix line of off-the-shelf power supply systems that allow batteries to be hot swappable, reducing or eliminating system downtime. The PowerBotix system also intelligently communicates with an onboard computer allowing for better mission



planning. Details may be found at the website listed.

"Because your mobile device never loses power to its internal CPU or other components, you could conceivably have it change its own batteries completely autonomously," stated Douglas Taylor, senior executive with rChordata.

Taylor also commented that PowerBotix allows you to quickly configure a system that is right for your application.

For further information, please contact:

rChordata, LLC	2100 Collingdale Place Charlotte, NC 28210 Website: **www.powerbotix.com**

180 Degree Servo Stretcher

With the Servo Stretecher from ServoCity, it's possibile to achieve 180° of rotation from just about any analog servo. The Servo Stretcher modifies the signal to the servo enabling it to rotate up to 90° either direction off of the center (neutral) position. It simply plugs in between the controlling source and the servo, just like a servo extension. The total amount of rotation is dependent on the type of radio control or servo controller. Results may be slightly less than 180° or slightly more. One of the features about the Servo Stretcher is users can limit each endpoint separately from the center position. If you only need the right to move 22 degrees and the left 88 degrees from center, no problem — just dial in the amount you need per side. You can even adjust the center (neutral) position separately from the endpoints.

The leads are constructed with heavy duty twisted wire and gold-plated connectors for super low resistance. The connectors are universal and can be used with any brand of receiver or controller. This product is not for use with servos that offer more than 90° rotation from the factory or any brand of digital servos. Dimensions are 2.4" x 0.86" x 0.47"; weight is 0.5 oz.

SPG400A-BM Servo Power Gearbox

ServoCity also introduces their line of bottom mount servo power gearboxes. The patented SPG400A-BM servo power gearbox is able to transform a standard size servo into a monster. This servo power gearbox will accept any standard size Hitec brand servo — digital or analog. The standard size servo is coupled to one of six optional gear ratios in order to maximize the torque output for use in the most demanding applications. An external potentiometer offers positioning feedback which allows the gearbox to perform accurate and repeatable motions. The structure is machined from high strength 6061 T-6 aluminum and utilizes dual ball bearings to support the 3/8" stainless steel shaft. The ABEC-5 precision ball bearings allow for smooth operation and provide support for up to a 200 lb load. Various attachments can easily be coupled to the aluminum hub by utilizing the 5-40 tapped holes. The SPG400A-BM can be purchased in kit form or as a fully assembled unit that is ready for use (units start at $59.99). Videos of this product in action are available at the website listed.

For further information on the previous two new products, please contact:

ServoCity	Website: **www.servocity.com**

Stencils for Quickturn and Full Feature PCB Prototypes

Sunstone Circuits® has enhanced their services by offering SMT stencils when placing an order within their Quickturn, Full Feature, and CAD Tool PCB123® product lines. The stencils join a growing suite of PCB solutions that include bundled assembly, PCB design software, and free DFM.

Stencils are used for even deposition of solder paste onto a bare circuit board. The use of stencils replaces hand soldering of surface-mount devices, and eliminates the inconsistencies often created by hand soldering. By adding stencils to a new or previous PCB order on the Sunstone website, a design engineer can increase the quality of their product and save valuable time.

All stencils feature the following:

- Permanent, non-removable, and non-fading fiducials.
- Exclusive performance-enhancing stencil design modifications, tailored specifically for each customer.
- 100% laser cut, ensuring the finest quality finish.
- All fixed frame stencils are double bonded to withstand extreme wear.
- Plates available with safety features to protect against sharp plate edges.

For further information, please contact:

Sunstone Circuits	Website: **www.sunstone.com**

Beetleweight Combat Robot Chassis

Kitbots introduces a new 3 lb Beetleweight combat robot chassis. Designed as the basis for a tough

Continued on page 55

bots IN BRIEF

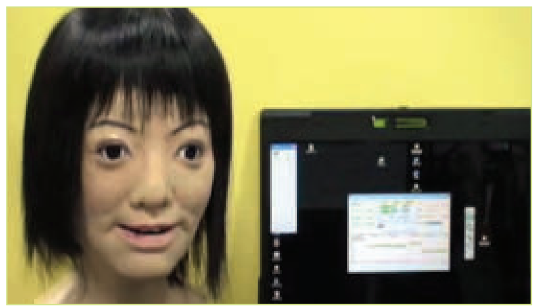
NEW ONE ARMED BANDIT

Willow Garage has announced the availability of an entirely new robot. Well, maybe not an *entirely* new robot. The PR2 SE is essentially the same as a PR2, except that (as you may have noticed from the picture) it's only got one arm.

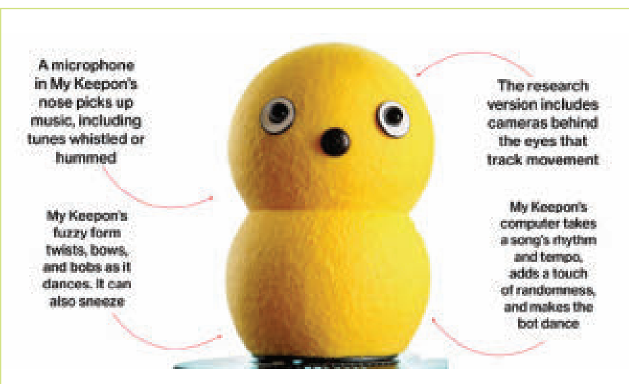
Despite having only half the armament of the original PR2, the SE boasts the same overall capabilities, along with an "updated sensor suite" that includes an integrated Microsoft Kinect. Lack of an entire arm may seem like a fairly significant issue for a robot, but many things that you can do with two arms you can also do with one — it just may take longer or require a bit more creativity.

If you do end up desperately needing another arm for your SE, you can buy one as an upgrade from Willlow Garage. Or, you can always build a slightly less fancy version on your own. Taking a big chunk out of the robot also takes a big chunk out of the price, which is the whole point of the SE version. The base price of the new PR2 SE is $285,000, and with Willow's 30 percent open source discount award, that comes down to just under $200,000. This is half the price of the fully armed and operational regular PR2 which costs $400,000 if you buy it straight up.

STOP WHILE YOU'RE AHEAD



Chyi-Yeu Lin and a team from the National Taiwan University of Science and Technology in Taipei created an eerie head that photographs a musical score with her camera/eyes, interprets the pitch, rhythm, and lyrics from an algorithm, then turns it into her version of music via synthesizer. Designed to someday be a restaurant receptionist, they need to perfect an equally creepy body to go with that head.



KEEPON DOING GOOD

Keepon has been around for almost four years and in that time, this bouncing bot has worked its magic into the hearts of autistic children, as well as 2,500,000 YouTube users. Toys 'R Us now has exclusive US rights and plans to bring him out to play in late October.

Keepon's story began about seven years ago with Hideki Kozima, a Japanese expert in artificial intelligence and robotics at the School of Project Design at Miyagi University. Kozima theorized that an emotive robot could help autistic children — who can be overwhelmed in face-to-face interactions — by reducing the complexities of communication to a few simple gestures. A child pats the robot on the head and it responds with a playful bob. The child talks to the robot and it turns to face him or her and nods.

To test his idea, Kozima then created Keepon — the fuzzy, mouthless robot packed with $30,000 worth of machinery, sensors, and computer chips. (The name is a portmanteau of the Japanese word for yellow, kiiroi, and the onomatopoeia pon for bounce.) In clinical use, a researcher in an observation room controls Keepon wirelessly, dictating its interactions with children. While testing the gizmo in day-care centers, Kozima found that autistic children made more eye contact with the robot than they did with people. Behaviors they rarely expressed toward humans — like touching and nurturing — became more commonplace. Since then, dozens of research centers and universities have bought the fuzzy bot for therapeutic work.

bots IN BRIEF

ANIMAL COMFORTS

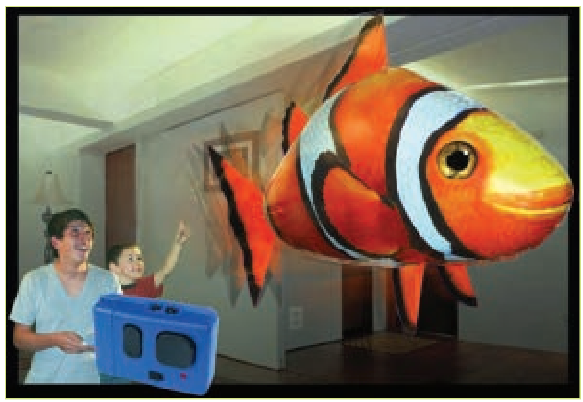
After the devastating catastrophe in Japan, Daiwa House donated two of its Paros to a previously abandoned retirement home in Suisyoen. Because real animal therapy is not always possible, residents can hold these substitutes and somehow find them comforting. While the sealbots would normally cost about $155 a month to lease, the generous company left Love and Peace for a two year stint.

Sitting only 27 km (17 miles) south of the stricken Fukushima Daiichi plant on a hill above an area ravaged by the tsunami, the Suisyoen retirement home is located in the middle of Japan's triple crises.

While the retirement home structure was spared major damage by the earthquake and subsequent tsunami, fears of radiation contamination from the nearby nuclear plant led officials to evacuate Suisyoen for two months until mid May.

AIR JAWS

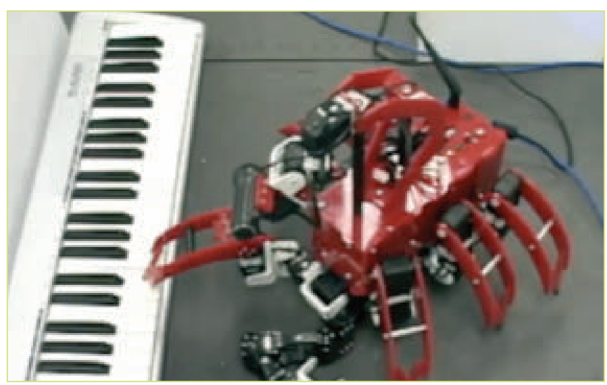
Take Jaws to a (literally) higher level with this IR remote controlled indoor Air Swimmer. Once the 57" long nylon shark is filled with helium, it can go 360° and move up, down, left, and right from up to 40 feet away. It can remain aloft for about two weeks at a time. The air swimmer also comes disguised as a Clownfish. Both will bounce off walls but be sure to remember to turn off your ceiling fan before use. Each swimmer requires seven AAA batteries (not included.)



KEYED UP HEXAPOD

This hexapod — Chiara — has certainly found a comfy little niche for itself in the robotic classical piano world by plonking away at some Beethoven.

Chiara itself is an open source educational robot developed by Carnegie Mellon University. It runs a free programming language called Tekkotsu, and this particular musical demo was put together by Ashwin Iyengar, a high school student.



FIJITY BFFs

Fijit Friends (that debuted at February's Toy Fair and are made by Mattel) can sing, dance, giggle, and tell jokes. With soft, tactile skin and an LED face, these friends respond to noise, movements, and tummy poking. Fijits react to over 30 specific voice commands, have about 150 phrases, and can interact with mobile apps, TV commercials, and other outside stimuli. When bedtime comes, they can become a nightlight.

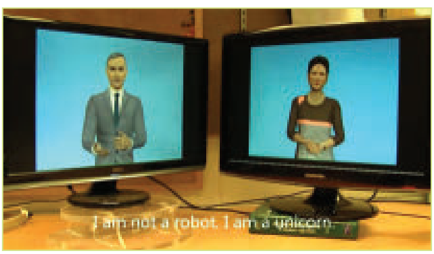
These little interactive robots were specifically designed for young girls from six years old and up. Since young girls seem to like to chat a lot, it's only natural that they are going to want someone to talk to but, of course, their human buddies can't be around them 24/7.

NOT WORKING OUT

Foxconn — an electronics manufacturer from Taiwan with huge factories in China — generates about 40 percent of the global consumer electronics revenue by creating things like iPhones and computer components on giant assembly lines staffed by humans. Until recently, you've probably never heard of Foxconn, but a series of worker suicides made people take a hard look at where our electronics were coming from. Foxconn has made some improvements (including nets around tall buildings), but by all accounts, the core of the problem (the work) remains "repetitive, exhausting, and alienating."

Foxconn has announced (at an employee dance party of all places) that they're planning on buying some robots to replace their human workforce. And by some robots, they mean one million robots over the next three years. So, for every one robot Foxconn currently has working at their manufacturing plants, they're going to buy a hundred more.

At this point, it's not sounding like Foxconn is trying to augment its human workforce with robots to make things easier on the humans. Foxconn employs something like 1.2 million people, and it's not too much of a stretch to imagine that one robot could probably work as efficiently as 1.2 humans. This could be a shift from "mostly human" to "mostly robot," with about a million jobs in the balance.



CHATTER BOXES

A chatbot is a computer program that's intended to fool people into thinking that it's human. Historically, this has been a tricky thing to do, and for the last 20 years there's been a $100,000 prize and gold medal waiting for the first computer program that can carry on a conversation that's indistinguishable from a human's. Arguably (very arguably), this could also be the first computer program to demonstrate an artificial intelligence.

Cornell's Creative Machines Lab decided to see what would happen if they put two chatbots face to virtual face and got them talking to one another. During the chat, things didn't go quite as crazy as might have been expected, but a fair amount of pointless argument, passive aggression, and random hilarity did ensue.

The 2011 Loebner Prize Competition in Artificial Intelligence will take place on October 19th, and if any of the entrant programs manage to fool two or more judges comparing two or more humans into thinking that it's a human, the program will win $25,000 and a silver medal. The final $100,000 prize will go to a program that includes a completely convincing audiovisual component as well, and that too may be closer than you think.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

WHAT REALLY MATTERS

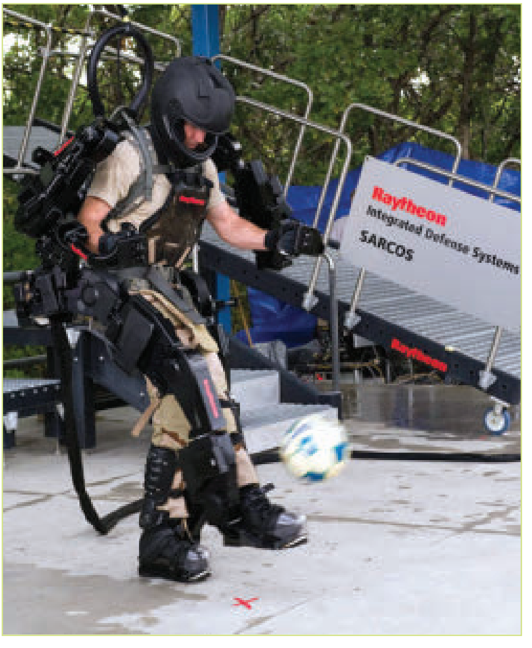
The go-to way of delivering medical supplies to rural areas of developing nations is to not deliver them at all, basically, forcing sick people to hike miles through mountains and jungles to get the drugs they need. If the weather's been bad and the roads are washed out, well, good luck. Solution? Do it all by air.

The only way to do that efficiently (or at all) is to scale it way down from planes and helicopters to small UAVs. This is the concept behind Matternet which seems to be both a technology and a company who wants to revolutionize the way medicine is delivered to the billion or so people who live completely cut off from road networks for at least part of the year. Matternet will be a network of autonomous quadrotor UAVs that use GPS and a beacon system to rapidly deliver small packages (containing drugs or medical testing supplies) to people who can't otherwise get them. Their first commercial platform (look for it in the next few months) will be able to fly 10 km while carrying a 2 kg load; it should be durable enough to make thousands of trips in variable weather. Plus, you get all this for only a few hundred dollars a unit. If it works out, Matternet could mean a drastic quality of life improvement for a lot of people.

Matternet will develop in three distinct phases: Phase 1 involves using a single UAV for point-to-point cargo transport. For example, a clinic uses a UAV to deliver drugs to an otherwise inaccessible nearby village in 30 minutes or less. Phase 2 will add remote, autonomous recharging stations to allow UAVs to juice up in between deliveries, enabling them to roam farther afield and make multiple deliveries without having to return to base. Connect the dots between base stations and you now have a delivery network. In Phase 3, all of these discrete networks grow large enough that they overlap, and it becomes possible to use a continuous chain of autonomously cooperating UAVs to transport things across entire continents very quickly and for cheap. Eventually, the idea is that Matternet turns into a sort of Internet for stuff, where you can make a request and get a physical object delivered to you. So, "Matter" net.

TALKING HEADS

This new robot is an evolution of the 2003 reading robot that was presented by the Korea's Electronic and Telecommunications Research Institute. This new model is now being used as an ambassador for the improvement of human-robot relations at the Daejon's National Science Museum. It will greet the visitors with nice and caring phrases of love, along with displaying its emotions through LED facial expressions.



IRON MAN HOMAGE

Sarcos recently said that its second-generation exoskeleton robot suit, XOS 2 — is now five years away from production. The wearable robotics suit augments the operator's strength by using a system of high pressure hydraulics, sensors, actuators, and controllers to bear the weight of an object while leaving its wearer agile enough to kick a soccer ball. It's also lighter, stronger, and more environmentally resistant, and it uses half the power of the company's first exoskeleton (XOS 1) which rolled out in 2008. The XOS 2 has been nicknamed the "Iron Man" suit in homage to the high tech power suit in the comics and movies.

The Sarcos exoskeleton first came out more than five years ago, when it was just a prototype developed as part of a DARPA program. Since then, Sarcos (now a division of US defense contractor, Raytheon) has significantly improved the device. The XOS 2 exoskeleton is designed to lighten a soldier's load and help the military reduce injuries. It also lets you pretend you're Tony Stark.

COMBAT ZONE

Featured This Month:

Features

26 *BUILD REPORT: Trilobite – a Tough Beetleweight Brick/Wedge* **by Pete Smith**

29 *BUILD REPORT: A Team Building Exercise* **by Pete Smith**

36 *PARTS IS PARTS: Fingertech 'Silver Spark' Gearmotor Review* **by Thomas Kenney**

38 *PARTS IS PARTS: Susie's Saga Continues – The Beginner's Guide to Motors* **by Morgan Berry**

40 *END OF THE ALKALINES* **by Pete Smith**

41 *CARTOON*

Events

31 *Upcoming and Completed*

32 *EVENT REPORT: Clash of the Bots 2* **by Pete Smith**

33 *EVENT REPORT: BOTBLAST Rocks Columbia Mall* **by Dave Graham**

BUILD REPORT:

Trilobite — a Tough Beetleweight Brick/Wedge

● by Pete Smith

I have long held the opinion that a good wedge/brick is the best type of bot for most beginners in the sport. Too many people try to build a complex bot with a weapon before having acquired the experience and knowledge required to make one work at all, let alone well. Often, they end up with a bot with an ineffective or non-functioning weapon on a slow weak chassis. The result is usually a boring shoving match if two such bots meet — or worse — the rapid and total destruction of that first bot when it meets one of the more seasoned teams. This experience is not likely to encourage them to stay in the sport!

The best first bot is probably also the cheapest: hack a RC toy. That's what my son and I did when he wanted his first bot way

FIGURE 1. CheepShot damage.

FIGURE 2. XI.

www.servomagazine.com/index.php?/magazine/article/october2011_CombatZone

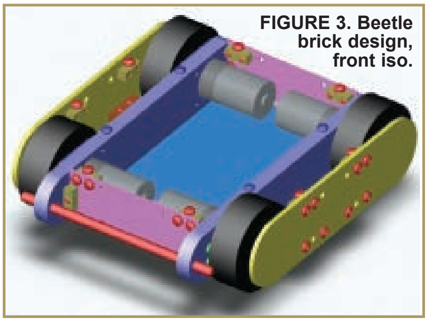


FIGURE 3. Beetle brick design, front iso.

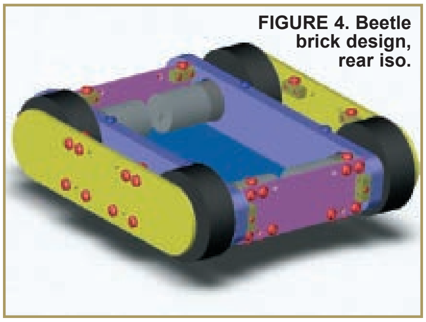


FIGURE 4. Beetle brick design, rear iso.

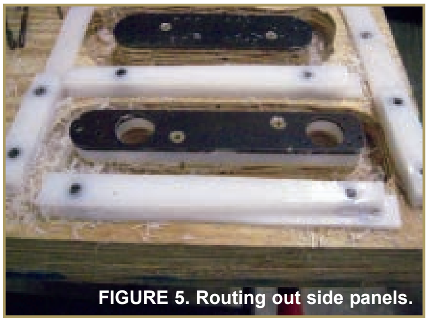


FIGURE 5. Routing out side panels.

back in 2003. We built a 12 lber called CheepShot (**Figure 1**). It never won a fight but was a "cheep" way to get my son competing, and soon led to requests for us to build something bigger and better. The new bot — a 30 lb class wedge/brick called Xhilarating impaX (**Figure 2**) — was the first robot I fully designed in CAD, and our first really successful design.

Eight years later, we have moved on to having only weaponed bots like Surgical Strike and Weta, God of Ugly Things, but those early bots were essential in making the steep learning curve in the hobby just a little smoother. I have designed and produced a range of weaponed bot kits over the last few years, but I felt an easier entry point to the sport was required: a 3 lb Beetleweight wedge/brick.

Brick/wedge bots are often accused of being boring, but that need not be the case. The first step in avoiding boredom is speed. A weaponless bot needs to be fast. There are two reasons for this. Firstly, slow is boring and since you have no weapon, you need to show aggression and take the fight to the opponent. If neither bot suffers much damage in the bout then you are more likely to win the judge's decision.

The second requirement is toughness. You need to be able to take the biggest hits and (not only keep working) show little visible damage. When a weapon blade strikes your bot, it applies an equal force on the bot that's doing the hitting. That's Newton's Third Law: For every action, there is an equal and opposite reaction. You can use that to your advantage but only if you can survive the hits yourself.

The third requirement is power. The best way to stop a weaponed bot is to push it into the arena wall, so your bot needs to have more power and traction than your opponent. You have the advantage in not having to put so much weight into a weapon system, so some can go towards a more powerful drive, and one that drives more wheels in contact with the ground.

The UHMW wall and 7075 aluminum panel design in Weta had proved so solid that I used the same structure in the new design (**Figure 3** and **Figure 4**). Designed using SolidWorks, two 3/8" thick UHMW walls would run the full length of the bot; these would be the major load bearing members. I decided to use the same 1,000 RPM motors that I use in Weta and mount them the same way. To keep the overall bot size down, I went with 2.25" wheels and planned to use 4S LiPo to keep the speed up despite the smaller wheels (Weta uses 3" wheels and 3S).

The wheel base was kept reasonably long since that helps the bot drive in a straight line and makes it easier to control. Too short a wheelbase makes a bot very tricky to drive fast. Too long and it can make turning on the spot hard on the motors and speed controllers — if it can turn at all.

The rear wheels are exposed at the back so that they will still make contact even if the nose of the bot is lifted up. This is useful if you are getting pushed by another wedge; you can still reverse quickly and turn aside to avoid the other bot. My previous wedge designs like Xhilarating impaX and CheepShot 3.0 both had the wheels fully protected, but were very vulnerable against wedges since once the nose was lifted up even a few degrees,


FIGURE 6. Trial build of Trilobite chassis, less the top.

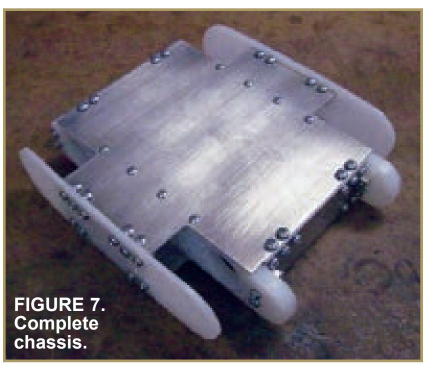


FIGURE 7. Complete chassis.

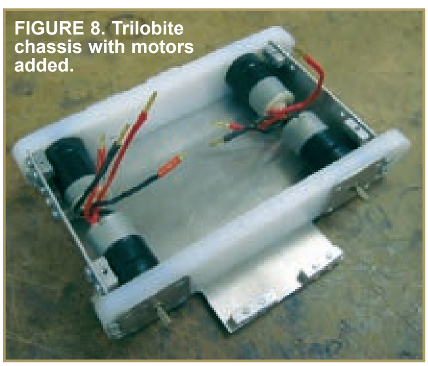


FIGURE 8. Trilobite chassis with motors added.

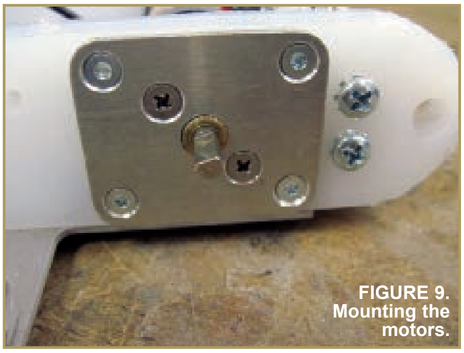


FIGURE 9. Mounting the motors.

FIGURE 10. Dave's hubs and Liteflite tires for Trilobite.

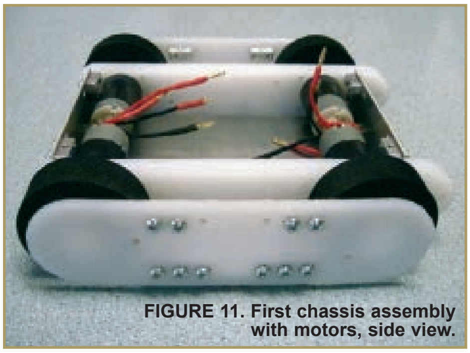


FIGURE 11. First chassis assembly with motors, side view.

the bot wheels lost contact with the ground and the bot was easily pushed about.

The UHMW and 1/16" thick 7075 panels are joined together using mini nutstrip and 6-32 screws, plus a few #6 x 1/2" Plastite screws are used between the bottom and the UHMW walls.

The top and bottom are the same panels that are on the front and rear bulkheads.

The top and bottom extend out past the wheels on either side so that 1/8" UHMW side armor can protect the wheels. This thickness had proved effective in Weta, so it should prove adequate for this application.

Two holes in the main UHMW walls are used for locating a 1/4" bar than will be used as the pivot for the front movable wedge. The thickness of the UHMW and titanium bar should prevent this hinge from being a weak point in the design.

I kept track of the weight of each part in an Excel spreadsheet. This is important because it's too easy to get your bot almost complete and find yourself overweight. Better to get it right from the start.

SolidWorks will work out the weight for you if you input the material's density, and it's worthwhile investing in an accurate electronic scale to weigh the other parts. If your budget does not allow for buying a scale, you can usually find one at your neighborhood post office or UPS agent. Ask politely at a quiet time and they are usually quite happy to let you use theirs.

Once I was happy with the design, I made dxf files of the aluminum panel and UHMW parts, and had a set of panels and templates water-cut by **www.teamwyhachi.com**.

They arrived within a week, and I set the templates up and routed out several sets of UHMW parts (**Figure 5**).

I had previously made all the required sections of nutstrip, cutting them to length on a chopsaw and then trimming to size on my mill. Putting all the chassis together for a trial build took only a matter of minutes (**Figure 6**). A cordless screwdriver comes in very handy here as there are a lot of screws!

Once all the panels are together, the holes for the Plastite screws can be drilled using the holes in the aluminum top and bottom panels as guides (**Figure 7**). The result is a remarkably strong and rigid chassis.

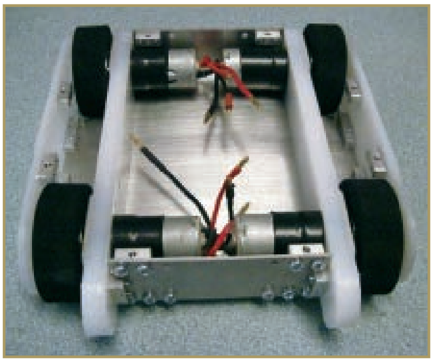
Stripping the chassis back down, I could then fit the drivemotors (**Figure 8**); each motor was secured in place using my standard "1000RPMMNT" mounting plates (**Figure 9**). Standard 4 mm "Dave Hubs" and 2.25" inch Liteflite wheels (**Figure 10**) — all from **www.robotmarketplace.com** — were added and the side armor refitted (**Figures 11 and 12**).

I originally intended to use one ESC per motor but when two of the four ESCs died shortly after installation and with time running out before the next event, I changed to using one of Banebots BB-12-45 per side with the two motors running in parallel. I had the ESCs prewired for use in Surgical Strike, so there was a lot of extra wire and weight over the four smaller ESCs but I had enough to spare for that. A standard BR6000 receiver (I mix for tank steering in the DX6 transmitter) and a Thunderpower 850 mAh 3S LiPo completed the wiring, and the bot was ready for its first drive.

Performance — even on 3S — was excellent, certainly fast enough for the smaller arenas, and the bot was easy to drive.

I used a holesaw to cut a large hole in the top panel to allow access to the battery connection, so this could be used to power the bot up for a fight. A strip of duct tape is used to keep the electronics in

FIGURE 12. First chassis assembly with motors, top front.



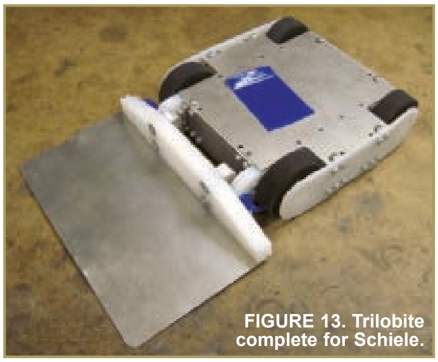


FIGURE 13. Trilobite complete for Schiele.

and shrapnel out.

There was only one day left before the bots debut at the Schiele Museum back in July, so I quickly put together a wedge using two chunks of 1/2" nutstrip, some UHMW, and a sheet of 1/16" 7075 aluminum. This was attached to the bot using a short length of 1/4" titanium rod. This was a tight fit in the holes and I thought it would hold up alright, but combat was to prove otherwise. The bot — now named Trilobite — was ready to go (**Figure 13**).

The bot performed reasonably well at the event. The wedge proved more a hindrance than a help as it kept getting stuck under the bumpers and the axle came loose. The bot was thrown about by both Weta and Grande Tambor but it suffered no more than a few scratches, A better wedge and some snowplow type attachments are needed, but I think it will perform well at its first big test at the Franklin Museum in October.

Kits of the chassis will be available from **www.kitbots.com** by the time you read this. I hope they help newbies get a good start in the sport. SV

BUILD REPORT:

A Team Building Exercise

● by Pete Smith

My Kitbots bot hockey team "Team Scotch Pies" had competed in one event and taken part in a couple of demonstrations, but the bots were retasked for a summer camp and were less than ideal. The bots were four wheel drive, but only used two cordless drill motors and they only weighed 8 lbs each (rather than the allowed 15 lbs). It was clear when they first met other custom-built hockey bots that they were simply outclassed.

A planned demonstration at the Durham Museum of Life and Science in March '11 gave me the impetus needed to build a new fleet of competitive bots.

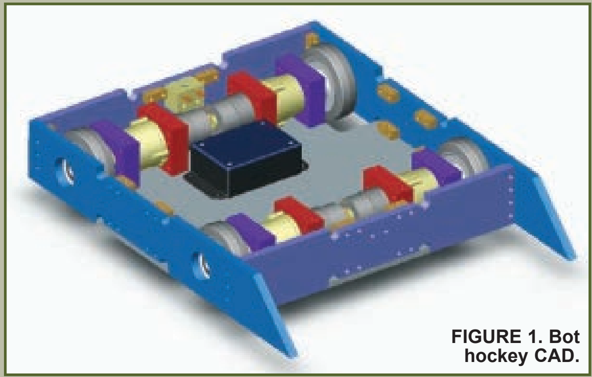


FIGURE 1. Bot hockey CAD.

To save time, I used as many standard Kitbots parts and familiar processes as I could. The finished design (**Figure 1**) uses template routed polycarbonate panels joined to together with my 3/8" nutstrip and four 18V cordless drill motors in the budget motor mounts, plus 3" Colsons with the standard hubs. The top and bottom are identical as are the two sides and the front and rear panels. This reduced the number of templates required and the work setting each one up. The top and bottom are 1/4" thick while the sides are 3/8".

The watercut templates were ordered from **www.team whyachi.com** and once they

FIGURE 2. Routing out the bot hockey top panel.

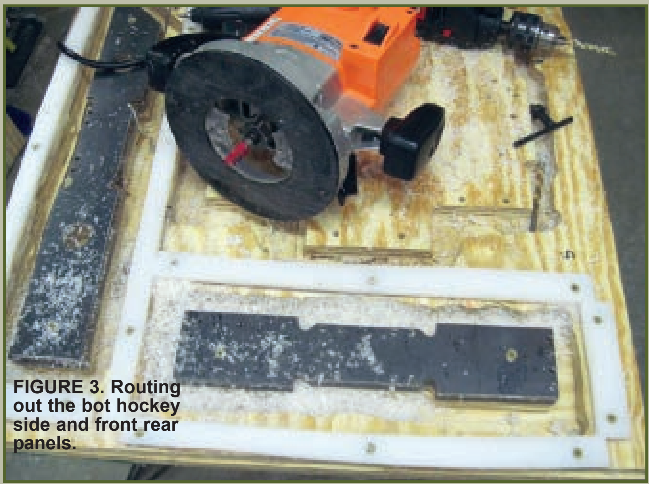


FIGURE 3. Routing out the bot hockey side and front rear panels.


FIGURE 4. Nutstrip getting cleaned up.

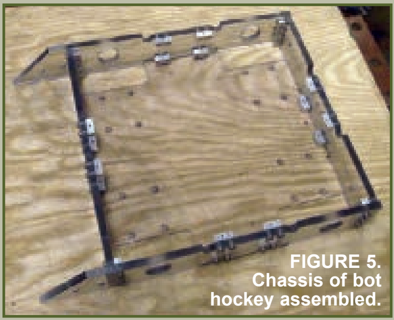


FIGURE 5. Chassis of bot hockey assembled.

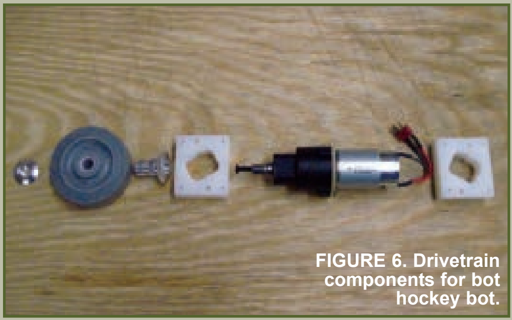


FIGURE 6. Drivetrain components for bot hockey bot.

arrived, I set them up pretty much as I described in an article in the March '10 issue of *SERVO*. I could quickly produce multiple copies of the top (**Figure 2**) and side panels (**Figure 3**). I found that it is necessary to cut the polycarbonate blanks close to the correct size with a jigsaw before routing to the final profile. If you try to route the full width of the cutter, it proves too much for the guiding bearing and it quickly fails. All the required screw holes are also in the templates, so they are drilled before the part is removed.

A bot hockey (**www.bothockey.com**) team consists of only three bots, but it is wise to have at least four so that you can swap out any bots that develop problems. The routing process allows one to mass-produce the parts easily, so making four bots is not that much more work than building a one-off design.

Sections of 3.8" nutstrip were cut off using a chopsaw and then trimmed to length using my mill (**Figure 4**). One can use a file to smooth each end or even leave them rough, but the mill makes quick work of it, especially when so many parts are required. The screw holes in the bottom panel were countersunk and then the chassis assembled (**Figure 5**) using Phillips head screws that have a locking patch (similar to McMaster part 96562A245) to ensure they do not vibrate loose.

Similar templates were used to produce the 32 separate motor mounting blocks which are used to fit the modified cordless drill motors and wheels. The parts required to produce just one drive assembly can be seen in **Figure 6** and the assemblies required for just one bot in **Figure 7**.

The drive assemblies are fitted to the baseplate (**Figure 8**) using four #10 Plastite screws, but one can use #10 sheet metal screws instead if the Plastite ones prove hard to find. The design allows them to be quickly replaced in the event of a failure since only very minimal disassembly is required, and each drive can be removed and replaced as a complete unit. The old bots would require at least 10 minutes' work to do what can be done in two in the new design. Reducing complexity also reduces the opportunity for mistakes to be made and this can be important in a competition environment.

I fitted Team Whyachi MS05 power on/off switches to each bot (**Figure 9**) so that they can easily be powered up and down without removing the covers. Since a special tool is required to operate them, it prevents unauthorized power-ups. This is useful at events where there are a lot of kids, some of whom — like me at that age — have difficulty stopping themselves from touching buttons!

Three of the bots were fitted with the latest Scorpion XXL ESCs from **www.robotpower.com** (on the left in **Figure 10**) and two 6S A123 battery packs in parallel. The fourth uses a pair of Victor 833s I had left over from our old 30 lb combat bot, together with 24V packs I assembled using the batteries from the dismantled cordless drills. These are much bigger and heavier than the A123 packs and have only half the capacity and twice the charge time, but since I was able to have three complete sets, this was not an issue. I found the two packs (about 2,400 mAH combined) lasted just long enough for the 10 minute matches without a noticeable drop in performance. The 24V does give a slight speed advantage over the

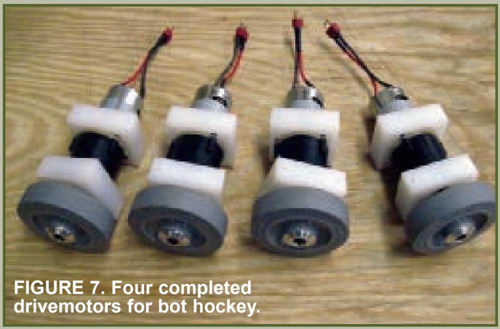


FIGURE 7. Four completed drivemotors for bot hockey.

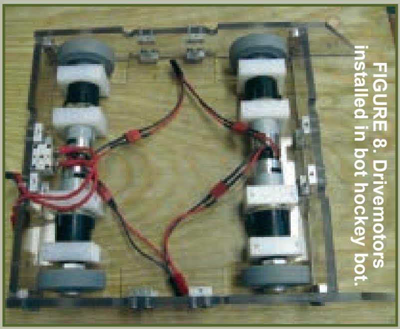


FIGURE 8. Drivemotors installed in bot hockey bot.

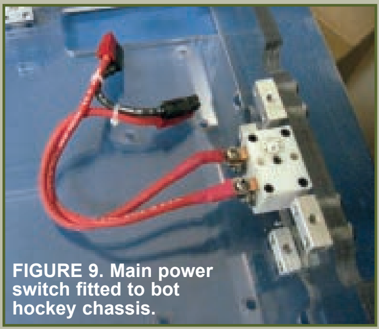


FIGURE 9. Main power switch fitted to bot hockey chassis.

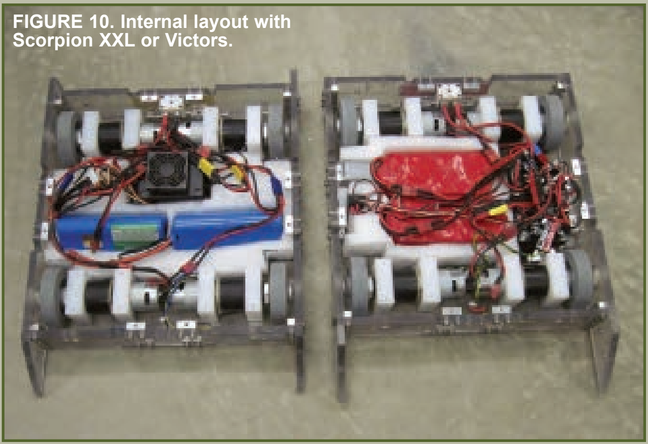


FIGURE 10. Internal layout with Scorpion XXL or Victors.

FIGURE 11. Team Scotch Pies with trophy.

19.8V of the A123s, but their poorer current sourcing ability makes them a pretty close match.

I also added a pair of color coded 10 mm LED "eyes" to the front of each bot so that it's easier to identify your bot in a melee (and because it just looks cool!). A 1K ohm resistor in series with each LED ensures good light output but also a long life at the voltages used in the bots.

The new team had its first real trial at the Schiele Museum Event in July '11 and they proved equal to the task. With a good driver, the bots proved to be more than a match for the competition and they can be seen a little tired but happy with their first place trophy in **Figure 11**.

Further events are in the planning stages and perhaps include a trip to RoboGames next year. SV

EVENTS

Completed and Upcoming Events

Completed Events for July-August 2011

Gulf Coast Robot Sports-8 was presented by Gulf Coast Robot Sports in Bradenton, FL on August 6th.

Schiele Museum Clash Of The Bots 2 was presented by Carolina Combat Robots in Gastonia, NC on July 23rd.

PA Bot Blast 2011 was presented by D.W. Robots in Bloomsburg, PA on July 16th.

Upcoming Events for October-November 2011

Franklin Institute 2011 will be presented by the North East Robotics Club in Philadelphia, PA on October 15th. Go to **www.nerc.us** for more information.

ComBots Cup VI will be presented by ComBots in San Mateo, CA on October 29-30. Go to **http://robogames.net/registration/event/view/11** for more information.

Mecha-Mayhem 2011 will be presented by the Chicago Robotic Combat Association in Rosemont, IL on October 22-23. Go to **www.thecrca.org** for more information. SV

EVENT REPORT: Clash of the Bots 2

● by Pete Smith

The Schiele Museum (**www.schielemuseum.org**) in Gastonia, NC held their second "Clash of the Bots" open day on Saturday, July 23rd. Carolina Combat (**www.carolina combat.com**) organized a Bot Hockey and Insect Class combat event as the main attraction.

The combat event had two Fairy weights, four Ants, and five Beetles entered, and three teams competed in Bot Hockey.

The venue is almost perfect. There was enough space for both the Bot Hockey and combat arenas, plus the pits in the main room. There was a separate space for charging LiPos and a rec room where the museum provided snacks, drinks, and lunch for the competitors.

The combat fights were round robin and there were some great matches including Weta versus Misdirected Aggression, and perhaps the best fight of all, Misdirected Aggression versus Grande Tambor (**Figure 1**). The fights drew standing room only crowds (**Figure 2**) and many can be seen on YouTube by searching "Clash of the Bots."


FIGURE 1. Misdirected Aggression vs Grande Tambor.

The Bot Hockey was much more competitive this year with Team Scotch Pies fielding new custom-built bots (**Figure 3**), last year's winners Team Pneusance, and a new entry for this year — Team Meatheads (**Figure 4**) — with two custom bot hockey bots and Hobbyweight combat bot Apollyon. Team Meatheads bots proved fast and powerful in their matches (**Figures 5 and 6**) but lacked the battery life and spare bots of the other teams, so while they held their own for the first half of the 10 minute matches, they quickly fell behind in the last minutes as their bots expired.

Team Pneusance and Team Scotch Pies both won three matches each and the latter won the tie breaker to get first place. Bot Hockey proved to be very popular with both the competitors and the crowds, including one kid from the audience who had played for a team last year and waited patiently for


FIGURE 2. Standing room only.

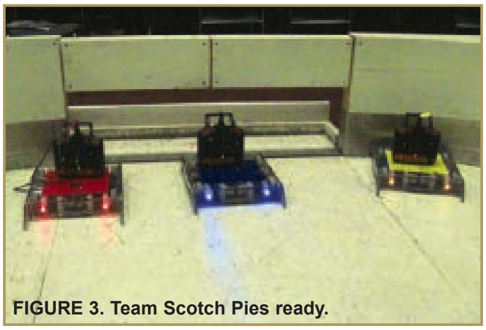


FIGURE 3. Team Scotch Pies ready.

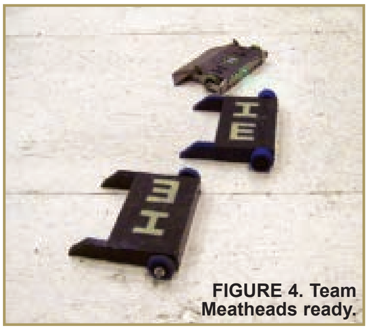


FIGURE 4. Team Meatheads ready.

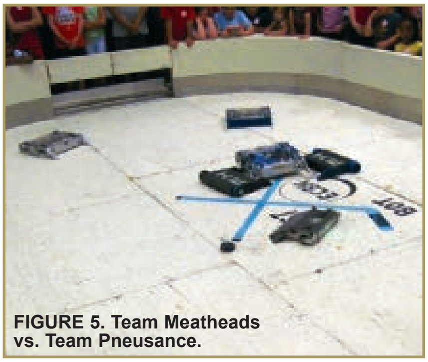


FIGURE 5. Team Meatheads vs. Team Pneusance.

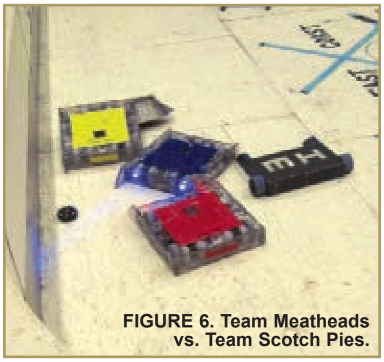


FIGURE 6. Team Meatheads vs. Team Scotch Pies.


FIGURE 7. Great big trophies.

two hours to get his chance again this time to play and score a goal.

The museum presented what were possibly the biggest trophies (**Figure 7**) ever awarded in robot combat, and $50 for each first place.

The event reportedly drew 560 paying spectators — double from the year before. **SV**

RESULTS

1st Place Fairyweights:	**Caterpillar**
1st Place Antweights:	**Gilbert**
1st Place Beetleweights:	**Weta, God of Ugly things**
1st Place Bot Hockey:	**Team Scotch Pies**

EVENT REPORT: BOTBLAST Rocks Columbia Mall

● by Dave Graham

The Columbia Mall in Bloomsburg, PA, hosted the fourth annual BotBlast competition on July 16th. Top fighting robot enthusiasts from Florida to Michigan took center stage at the mall (**Figure 1**) and brought 31 of the most destructive Insect class robots ever assembled for a BotBlast competition.

Event organizer Jeremy Campbell and Team Dreadfully Wicked Robots welcomed six 150 gram Flea (a.k.a., Fairy) class robots, eight one pound hungry Ant bots, and 12 three pound voracious Beetles. Campbell made one change to this year's competition, deleting the 12 pound Hobbyweight class and adding the six pound Mantisweight class to the venue. The new Mantis class drew five competitors.

The Flea competition featured six bots, four with spinning weapons. In first round action, I drove my bot, Hedgehog (**Figure 2**), and Team Mateo to a win over Jamison Go and his bot, P150 (**Figure 3**). During the match, the two Fleas collided and Hedgehog sliced the tire off P150. Hedgehog went undefeated in the winner's bracket to the final match, where he faced teammate Matt Benjamin, driving Baby V (**Figure 4**). Hedgehog took the gold and also won the Fleaweight rumble.

The Ant competition pitted some nasty spinners against an outnumbered group of wedges. In second round action, Andy Hoffman-Patalona's horizontal spinner, Box #5 (**Figure 5**), took out Kyle Singer and Fangus Ultimate (**Figure 6**) in a close match. Then, Chris Atwood's wedge bot Antelope (**Figure 7**) sent Box #5 to the loser's bracket and moved on to the Antweight final match.

The loser's bracket pitted Box #5 against Fangus Ultimate in a rematch for a chance to go to the final match against Antelope. This time, Fangus Ultimate sent Box #5 packing, and squared off with Antelope in the final.

The final match was back and forth with lots of sparks as Fangus Ultimate's spinning blade worked on Antelope's steel front end. Eventually, Fangus Ultimate knocked a wheel off of Antelope winning the match, and forcing a second match in the double-elimination format. The second match went the

FIGURE 1. Group shot of the competitors at Columbia Mall.

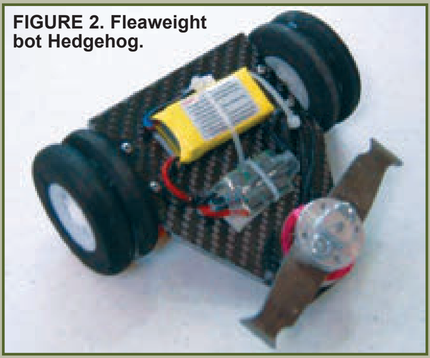


FIGURE 2. Fleaweight bot Hedgehog.

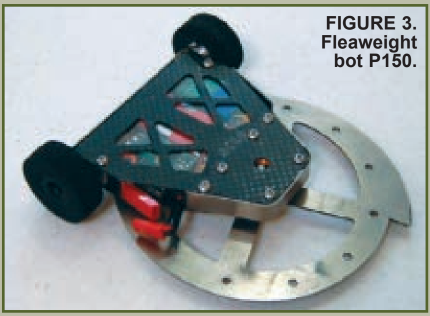


FIGURE 3. Fleaweight bot P150.

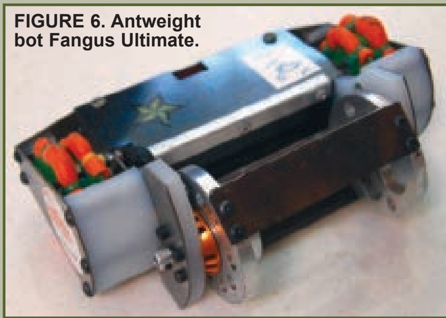


FIGURE 6. Antweight bot Fangus Ultimate.

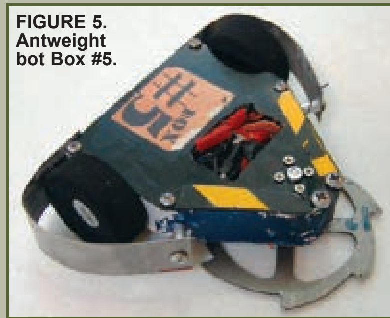


FIGURE 5. Antweight bot Box #5.


FIGURE 4. Fleaweight bot Baby V.


FIGURE 7. Antweight bot Antelope.

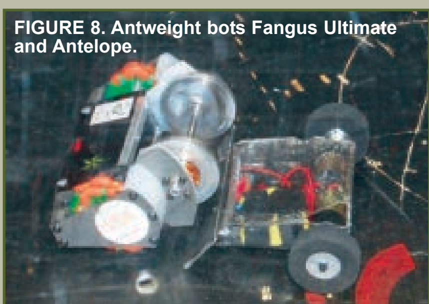


FIGURE 8. Antweight bots Fangus Ultimate and Antelope.

FIGURE 9. Beetleweight bot One Fierce Lawn Boy.

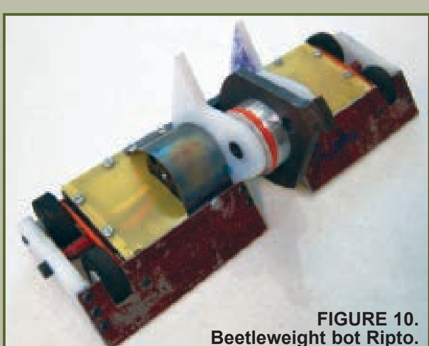


FIGURE 10. Beetleweight bot Ripto.

distance as Antelope's steel front plate was effective and all Fangus Ultimate could do was repeatedly ride up on Antelope's front end (**Figure 8**). Fangus Ultimate never got a "bite" of Antelope and as a result, the judges gave the nod and the Antweight title to Antelope. Matt Benjamin, driving team namesake bot, Mateo, won the Ant rumble.

The Beetleweight class definitely had the most twists and turns in the story line. Gene Burbeck of Team Fierce Robots and his creation, One Fierce Lawn Boy (**Figure 9**), defeated always tough Kyle Singer and his bot Ripto (**Figure 10**) in third round action. One Fierce Lawn Boy then took out Jamison Go's bot, Cake (**Figure 11**), in the fourth round of the winner's bracket to make it to the final match.

Cake's weapon is a spinning steel drum, and is unique in that both the brushless drive motor and the electronic speed controller are housed inside the spinning drum. That makes for an effective weapon, as well as some interesting gyro dance moves.

In an interesting turn of events, lone female competitor Moraima Ortiz, Team Danger Zone, and her creation Foofie (**Figure 12** — note the matching flowers in her hair), suffered what appeared to be a career-ending injury at the hands of Cake in first round action, but came back and made it to the fourth round of the loser's bracket before being eliminated by Ripto. Along the way, Foofie went up against Evan Steeves and his bot Hexxus (**Figure 13**). Steeves, in a true show of sportsmanship, decided not to spin-up the weapon on Hexxus for his match with Foofie, and Foofie promptly stuck him on the wall where Hexxus was counted out by the judges for no movement! It was the upset of the day.

Steeves had a tough day, but was named BotBlast "Rookie of the Year" by event organizer Campbell. The Beetleweight class also featured the event's youngest competitor — Brandon Young — shown getting some help from his dad before a match (**Figure 14**).

The final bout in the Beetle

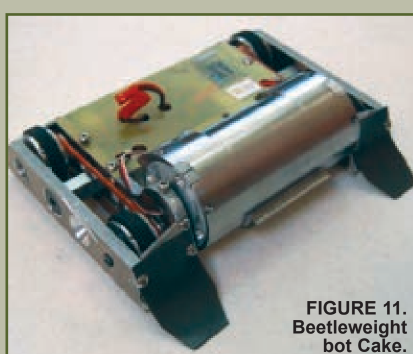


FIGURE 11. Beetleweight bot Cake.

FIGURE 12. Moraima Ortiz and her Beetleweight bot Foofie.

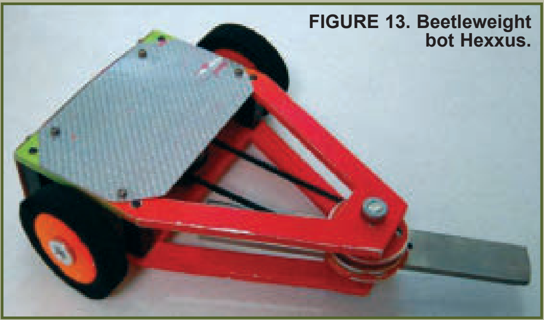


FIGURE 13. Beetleweight bot Hexxus.

FIGURE 14. Youngest competitor Brandon Young and his dad.

FIGURE 15. Beetleweight bots Ripto and Cake in the arena.

Loser's bracket saw the two bots eliminated from the Winner's bracket by One Fierce Lawn Boy — Ripto and Cake — fighting for the chance to face One Fierce Lawn Boy again in the final match. The two bots spun up their weapons (**Figure 15**) for what was to be a match-ending hit when Cake launched Ripto, and Ripto came to rest stuck between the arena bumper and the Lexan arena wall (**Figure 16**). The hit damaged Cake's spinning drum (**Figure 17**), causing Jamison Go to replace the drum prior to the Beetle championship match. One Fierce Lawn Boy defeated Cake in the final match to take first place. Ripto did come back to win the Beetle rumble, and builder/driver Kyle Singer was voted the "Best Driver" in the competition.

I'd like to add one more special award as the writer of this article — and that award goes to the "Best Pit Crew." Clearly Jamison Go wins this one. Jamison rebuilt multiple bots multiple times, and was competitive in the Flea, Ant, and Beetleweight classes. He did a yeoman's job of keeping his bots in every match and answering every bell. Well done, Jamison.

The Mantis competition showcased five bots, including special award winner Reclipso (**Figure 18**), winner of the "Coolest Robot" award. Reclipso is the creative design of Zac O'Donnell, and sports a lifting arm operated by a brushless motor driving a camshaft that is engaged by a solenoid to violently jerk the lifting arm upward. While voted the Coolest Robot, Reclipso could only activate the lifting arm once, and failed to demonstrate the "flipping power" required to be competitive.

Nonetheless, Reclipso made it through the round-robin competition to face Gene Burbeck and his newest monster, One Fierce Bushwhacker (**Figure 19**), in the Mantisweight final match. Burbeck and his One Fierce Bushwhacker dominated the match and the Mantisweight class, going undefeated. The Mantis Rumble was won by Brandon Young and his bot, Catapult.

Gene Burbeck and his bots, One

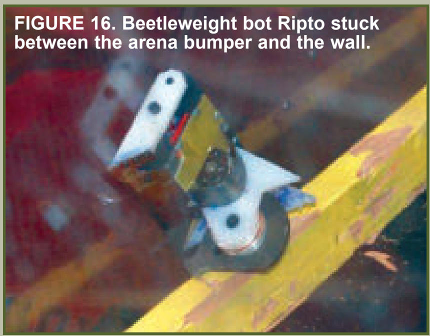


FIGURE 16. Beetleweight bot Ripto stuck between the arena bumper and the wall.

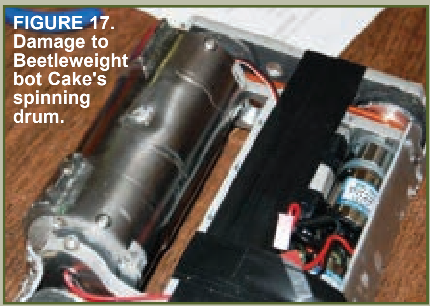


FIGURE 17. Damage to Beetleweight bot Cake's spinning drum.

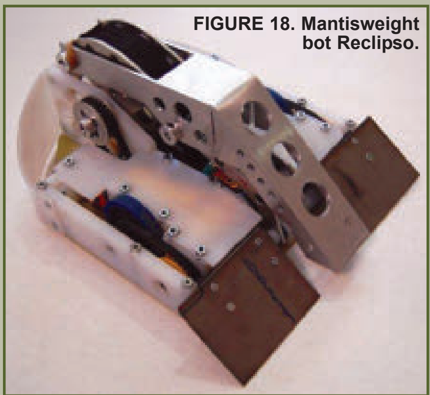


FIGURE 18. Mantisweight bot Reclipso.

FIGURE 19. Mantisweight bot One Fierce Bushwhacker.

TABLE 1 — WINNERS.

	FLEA	ANT	BEETLE	MANTIS
1st:	Hedgehog *Dave Graham*	Anelope *Chris Atwood*	One Fierce Lawn Boy *Gene Burbeck*	One Fierce Bushwhacker *Gene Burbeck*
2nd:	Baby V *Matt Benjamin*	Fangus Ultimate *Kyle Singer*	Cake *Jamison Go*	Reclipso *Zac O'Donnell*
3rd:	Rebound *Jeremy Campbell*	Box #5 *Andy Hoffman-Patalona*	Ripto *Kyle Singer*	Dead Metal *Chris Atwood*
Rumble:	Hedgehog *Dave Graham*	Mateo *Matt Benjamin*	Ripto *Kyle Singer*	Catapult *Brandon Young*

FIGURE 20. Gene Burbeck, Team Fierce Robots, and his Beetleweight bot One Fierce Lawn Boy and Mantisweight bot One Fierce Bushwhacker.

Fierce Lawn Boy and One Fierce Bushwhacker (**Figure 20**), took home three special awards: Best Engineered Robot, One Fierce Lawn Boy; Longest Distance Traveled, Michigan; and Most Destructive Robot, anything with "One Fierce" in the name. Zac O'Donnell received the BotBlast "Sportsmanship" award.

TABLE 2 — SPECIAL AWARD WINNERS.

- Longest Distance Traveled: ***Gene Burbeck***
- Rookie of the Year: ***Evan Steeves***
- Sportsmanship: ***Zac O'Donnell***
- Best Driver: ***Kyle Singer***
- Best Engineered: ***One Fierce Lawn Boy***
- Coolest: ***Reclipso***
- Most Destructive: ***Anything starting with "One Fierce" in the name***
- Author's Award for Best Pit Crew: ***Jamison Go***

A list of the winners is shown in **Table 1**, and a list of the special award winners as either voted on by the competitors or selected by the event organizer, is shown in **Table 2**. The winners shared plenty of prizes, including custom designed sequenced starting lights for first place, trophies and plaques for second place, third place, rumble winners, and special awards, along with tools from mall merchants and food gift cards from local vendors. There were also door prizes for registered competitors (**Figure 21**).

Event organizer Campbell credits his family (**Figure 22)** for making BotBlast 2011 an overwhelming success. While Jeremy (second from the right) is the ring leader, grandmother Margaret Sponenberg (far left) sponsors all the winner trophies and plaques; mom, Trish (second from the left) takes care of registration, voting, and supervises the pit area; and dad, Warren (far right) is the best arena man in the sport, taking care of virtually every aspect of the arena, down to the artwork on the arena floor.

Mark your calendar now for mid-July 2012, and plan to attend this fun event! You can follow BotBlast on their website at **www.botblast.webs.com**. SV

FIGURE 21. Awards table.

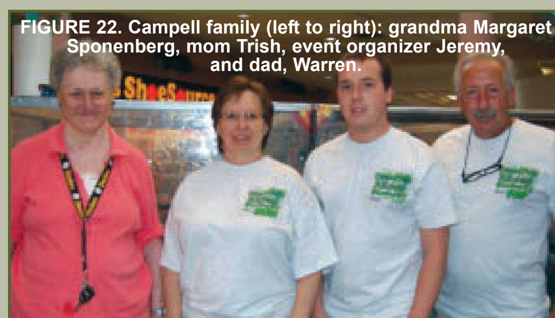


FIGURE 22. Campell family (left to right): grandma Margaret Sponenberg, mom Trish, event organizer Jeremy, and dad, Warren.

PARTS IS PARTS: Fingertech 'Silver Spark' Gearmotor Review

● by Thomas Kenney

FIGURE 1.

Fingertech Robotics' first line of Insect-sized gearmotors — the Gold Sparks — were originally released in 2009 as a drop-in replacement for the commonly used Banebots 16 mm models. Popular among Antweight robots for their high power/weight ratio, convenience, and availability of multiple gear ratios, the builder community felt the Banebots motors needed more reliability. The Gold Sparks fixed most of the issues, completely replicating

the design for the most part, while substituting more reliable materials to strengthen the design where needed.

Despite the improvements, there were still some weak points in the Gold Sparks gearmotor, mainly gears occasionally stripping out and breaking due to the small pitch, along with the output stage breaking loose of the drive shaft. Built to combat these issues, the main feature of Fingertech's newer 'Silver Spark' motors are the larger pitch of the gears (**Figure 2**) compared to the Gold Sparks, while keeping the same basic dimensions, 11 mm mounting pattern, and not adding more than a few grams for the equivalent gear ratio. In addition to all of this, they've added a whole new batch of gear ratios, ranging from half of the lowest Gold Spark ratio to twice its highest.

In the months leading up to their release, Fingertech has had a number of these motors going through rigorous testing in the arena. I've tried out a number of the new models in some of my recent small bots. In August '11, I wrote a build report for *SERVO* on one of these bots — an Antweight wedge. I had been running surplus Maxon 17:1 gearmotors for years, and with the Maxon stock starting to dwindle, the lower ratio Sparks now available, and the FK-050's ability to run off of 18.5V without burning up, it seemed the perfect opportunity to switch over and begin looking for an alternative drive motor. There are more details on the build itself in the mentioned article. Overall, the 11.1:1 Silver Sparks have handled more abuse over a dozen fights that it's taken to kill at least one Maxon motor in the same time frame.

A few months before building Rudy, I first tried out the Silver Sparks in a much more abusive testing application, namely using them as the drive for my three pound weaponed robot, Misdirected Aggression. I was stepping down from the B16 motors that were usually the standard for a Beetle spinner's drive train in an attempt to put some more weight into the armor. I had already tested some 50:1 Gold Sparks in the application, and they had stripped out after a few seconds of driving around, so I wasn't too confident in the newer gearmotors at first.

Despite this, the final Silver models eventually proved their worth, and held up fine until a massive impact ripped the motor from the gearhead, despite the red-Loctited mounting screws. This issue was eventually fixed by press-mounting the full gearmotor in a 1" wide UHMW block (**Figure 3**) similar to how I had previously secured the B16s.

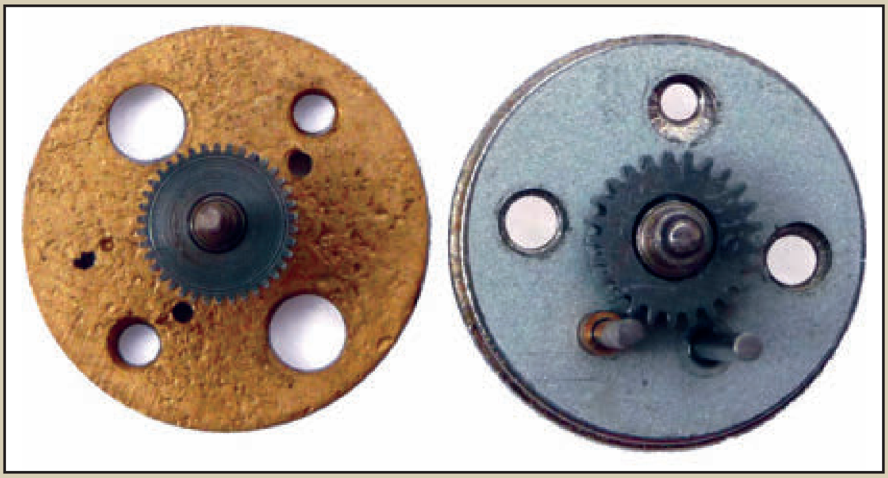


FIGURE 2. The final stage of the Silver Spark gearbox compared to the smaller pitch of its predecessor (left).

In addition to eliminating the need for mounting screws to the gearmotor's face, this press-fit supported the FK-050 motor, preventing it from being removed from the gearhead. The Silver Sparks have worked great from then onward, with no motor or gearbox failures not related to the bot's unprotected wheels. I'd recommend a similar mounting method for anyone planning on using a single pair of Silver Sparks in anything over one pound.

To conclude, the Silver Spark gearmotors have proven their value, holding together in applications where most similar small gearmotors would have failed. The Silver Spark gearmotors are currently available from **www.fingertechrobotics.com**, and FingerTech's US distributer at **www.robotmarketplace.com**. SV

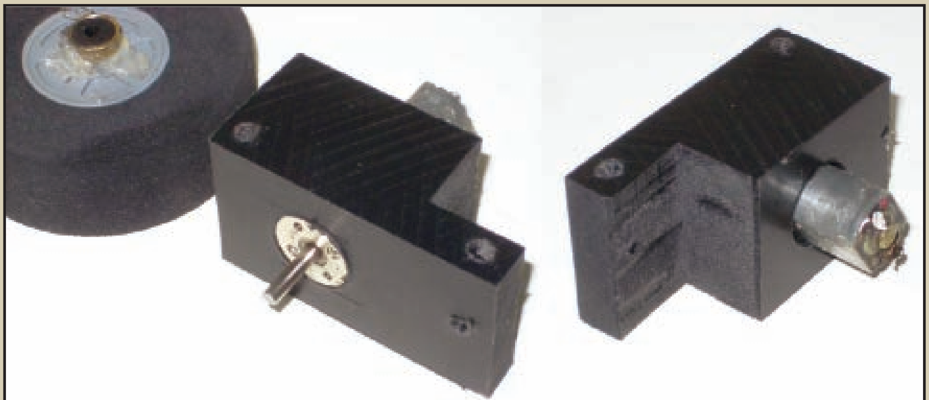


FIGURE 3. Supportive mounting for a 3 lb Beetleweight's Spark drive motor.

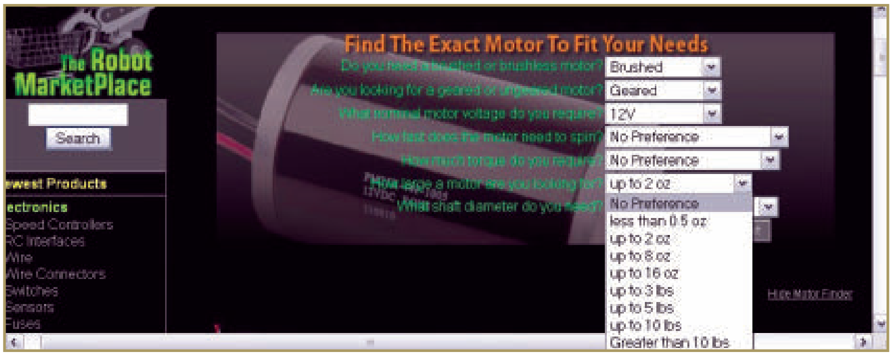
PARTS IS PARTS: Susie's Saga Continues – The Beginner's Guide to Motors

● by Morgan Berry

In the last issue of *SERVO*, we introduced you to Susie — a young girl trying to begin a career in combat robotics. We've had quite a few inquiries into the fate of this young robotics heroine, and we are happy to report that the information Susie learned about gears was the first stepping stone on her path to combat robotic greatness. In other words, Susie has been kicking some serious robot butt since she completed her first robot: The Destroyer! But now, after a year of victory upon victory in the local competitions, Susie has her eyes set on a big, national level youth robotics contest two weeks away, and she realized her little Destroyer isn't going to cut it for the fierce competition she will face.

She decides to rebuild Destroyer, but due to time constraints she can't possibly change everything she would like. What should Susie focus on to give her bot the extra edge against all those other robotics prodigies? After examining her bot closely, her answer is obvious.

The motor she bought from her local hobby shop bargain bin is just downright puny. She's geared the motor to increase her torque as much as she can, but it's not enough. All the best driving skills in the world can't make up for a weak motor. Susie immediately sits down at the computer and begins researching motors.



One of the older competitors in Susie's league, Bobby, learns of his friend's plan to upgrade her motor and tells her about a great tool that helped him out the last time he bought a motor. The Robot Marketplace — one of the many online stores for robot lovers — has a search engine to help determine what kind of motor a bot needs. This seems like an obvious place for Susie to start, so she logs on and scrolls through the questions: "Does your motor need to be brushed or brushless?" "What nominal motor voltage do you require?" "What is the maximum current the motor should draw?" Susie's eyes widen; she doesn't even begin to know how to answer the questions. So — like any good scientist — Susie starts at the beginning.

The first question on the list is brushed or brushless. Susie does some research and learns that DC motors typically have an electrical switch called a commutator which changes the direction of the current in the motor in order to create the rotating force, or torque. In a brushed motor, a metal brush is used to contact the surface of the commutator and create a current. A brushless motor replaces this metal brush with an electrical system, making it more efficient and less susceptible to wear than a brushed motor.

Brushless motors are substantially more expensive than brushed, provide a monstrous amount of RPM, and use a different type of controller. Some also think the electrical system is not as hardy as the more basic mechanical brushes. Susie decides that brushless motors are too involved for a quick build and — for the time being at least — she will stick with a brushed motor.

The next question Susie needs to answer is whether her motor will be geared or gearless. Geared motors — also known as gearmotor's — come with the gears needed already built in; they are a package deal and take a lot of the work of figuring out ratios and the other factors involved with gears. Gearless motors obviously do not

MOTOR TYPE	RPM	TIME TO TRAVEL 10 FT. WITH TWO INCH WHEELS	TIME TO TRAVEL 10 FT. WITH ONE INCH WHEELS
COPAL SH50 50:1 Gearmotor	300 (at 12V)	4.6 seconds	7.7 seconds
COPAL 50:1 Gearmotor – 12VDC	330	3.5 seconds	7.0 seconds
COPAL 50:1 Gearmotor – 6V	500	2.3 seconds	4.6 seconds
COPAL 30:1 Gearmotor	770	1.5 seconds	3.0 seconds
COPAL 60:1 Gearmotor	410	2.8 seconds	5.6 seconds

have these systems and require purchasing separate ones.

Although Susie already has a gear system that she worked hard to develop (*SERVO*, September '11), the system is specific to her old, less powerful motor and rather than changing the motor and the gears in such a short time period, Susie decides that this situation calls for a motor with the gearbox included.

The next questions regard the voltage, RPM (speed), and torque of the gearmotor. These factors are all dependent on each other. For a given motor, RPM and torque are opposing factors; the higher the RPM, the lower the torque, and vice versa. While voltage increases both these factors overall, it is constrained by the increase in the weight of the battery that accompanies higher voltage.

Antweights typically run in the 6V to 18V range. Since RPM and torque are dependent on each other, it's hard to exactly pin down the balance needed, so Susie decides to examine each specific motor and see what fits her needs best; she leaves these sections labeled "no preference." However, to get a rough idea of the ranges she is looking for, Susie sits down to do some calculations.

Susie has two inch diameter wheels, so she calculates the circumference of her wheels to be 6.28 in. An Antweight arena is usually between four and eight feet across. So, at 100 RPM (or 1.6 rotations per second) her bot can travel 10.048 inches in one second; it will take the bot 4.77 seconds to travel across the minimum sized arena. That's awfully slow, especially since the RPM does not take into account that the differing tractions of various arenas might influence speed. So, Susie knows she must have a significantly higher RPM for her bot.

A rule of thumb for a pusher bot is that the bot should be able to push three to five times its weight; for the 1 lb Antweight, this is 3-5 lb. How does this relate to the torque? Since torque is measured in a unit that compares a weight measurement to a distance measurement (in this case, ounce-inches), the key is to calculate the length of the lever arm the torque is being applied to, or in Susie's case, the radius of the wheel.

So, Susie takes the one inch radius of her wheel and multiplies it by a hypothetical torque (50 oz-in), then determines that the bot can push 50 oz, or 3.12 times the weight of her bot. Keeping in mind that she'll need a second motor for the second wheel she needs to spin, this increases the number to 6.24 times the weight of her bot. In other words, this will give Susie quite a powerful little bot, and she aims to look for a motor with a torque around this area.

Susie selects "up to 2 oz" for the size of the motor because anything larger will be too heavy for her 1 lb Antweight. She clicks "Find a match" and begins to scroll through the results. She rules out a 6V motor with a torque of 103.2 oz-in and 23 RPM; this is much too slow for her needs. She also quickly negates another with a 600 RPM; the torque is too low. Finally, she spots a line of Copal motors that seem to be what she is looking for. Copal is a very popular brand for Antweight bots. The only problem is that each motor involves a different gear ratio.

Susie is frustrated; she thought that by selecting a geared motor she could leave this entire portion out. But, keeping in mind what she previously learned about gears, she has a suspicion she needs a higher gear ratio, and clicks the motor with the 60:1 ratio. Sure enough, the RPM is 410 at 12V, which is about half of the 30:1 motor; the torque is 71.44 oz-in, about double the torque of the 30:1 motor. These are some awesome specs! Susie's bot will have the power to push nine times its own weight, and can travel three feet in one second. At .88 oz each, the two motors will only take up a fraction of her weight limit, and at 12 volts, the battery required for the motor is reasonably light, as well.

The only downside: The motors are about $20 each, so Susie will have to dip far into her babysitting earnings to afford them. Since this is the only purchase Susie will need to make for her new and improved bot, however, she decides to spring for the expensive motors. She is confident that her motor will make the Destroyer 2.0 a fierce competitor in her upcoming competition, but realizing that a good motor is only one piece of the puzzle, she resolves to work hard on improving her driving skills so the expense she puts into the motor won't be in vain. **SV**

End Of the Alkalines

● by Pete Smith

I wrote an article about programming cheap 2.4 GHz HK-T6A transmitters (**Figure 1**) that appeared in the January'11 issue of *SERVO*. In that, I commented on how it's probably a good idea to convert from the standard AA alkaline batteries to rechargeable batteries. I use the radios with four bot hockey bots, and two demonstration days and a week's summer robot camp have eaten up numerous sets of disposable batteries. A competition in July was the impetus to get the radios converted to using the more environmentally friendly and ultimately cheaper NiMH (Nickel Manganese Hydride) rechargeable batteries.

The various manufacturers use different types of charging plugs and polarities for their radios, so it's important that one uses the right type. I have an old Futuba TX and the charger had the right size plug and the same polarity on the contacts as that on the HK-T6A, so I knew it would be easy to get the wall charger units to match.

There are two ways to add rechargeable batteries. The first and easiest is to simply fit eight rechargeable AA sized NiMh cells and plug in a Futuba style charger. This has the advantage that you do not have to open the radio or make any changes to it. The disadvantage is that there are many interconnections between the cells, and a bad connection due to a loose battery clip could result in the radio losing power and you losing a fight or a game of bot hockey. I had already experienced loose batteries in one of the radios, so I decided to go with the second more complex — but ultimately more reliable — solution and the one more expensive radios use: a single 9.6V NiMH battery pack.

There are many of these available on the market and almost all would do the job, but I wanted a high capacity pack and one for which I could get the overall dimensions.

The one I chose (**Figure 2**) was from **www.onlybatterypacks.com**. It was a 9.6V NiMH 2,500 mAh pack that has the dimensions to fit the battery compartment in the HK-T6A radio. I had previously bought packs from them for my two Spektrum radios and they had performed well. A battery pack of this size costs about $26.

It is necessary to open up the radio in order to gain access to the connecter that hooks up the batteries to the rest of the electronics. There are four screws on the back (**Figure 3**) that hold the two halves of the radio together. Remove them and then unplug the connector circled in red (**Figure 4**) and cut the black and red wires where indicated by the blue cross.

Next, remove and discard all the metal battery contacts from the back half of the radio using a pair of pliers. These came off easily on all four of my radios.

FIGURE 1.

The battery was a little larger than advertised and could not be fully seated in the compartment, so I trimmed off the plastic moldings that held the contacts (**Figure 5**) with a good box cutter. *DO NOT REMOVE THE ONE INDICATED IN RED*. This one is used as the catch for the cover of the battery compartment. As can be seen, I took a small chunk out of this one by mistake, but it still works.

Next, I soldered the leads on the battery pack to the connector from the radio (**Figure 6**) and protected the joints with heatshrink tubing. I had tried to find what — if any — commercial radios used the same connectors but was unsuccessful, so the packs I bought had no connectors at a slightly reduced price.

Feed the connector through the opening at the top left of the

FIGURE 2.

FIGURE 3.

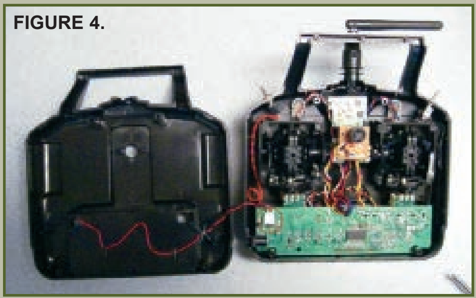


FIGURE 4.

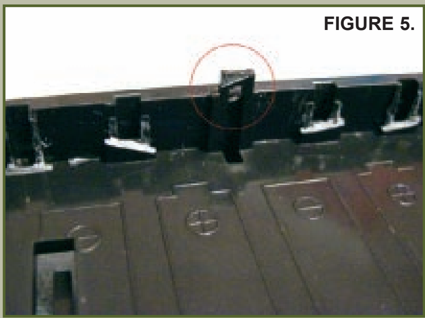


FIGURE 5.

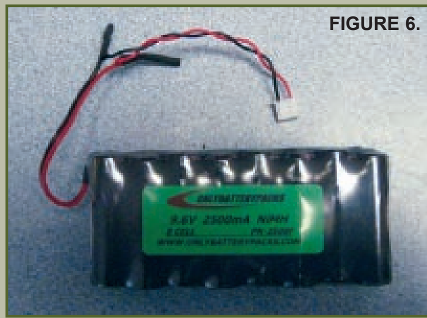


FIGURE 6.


FIGURE 7.

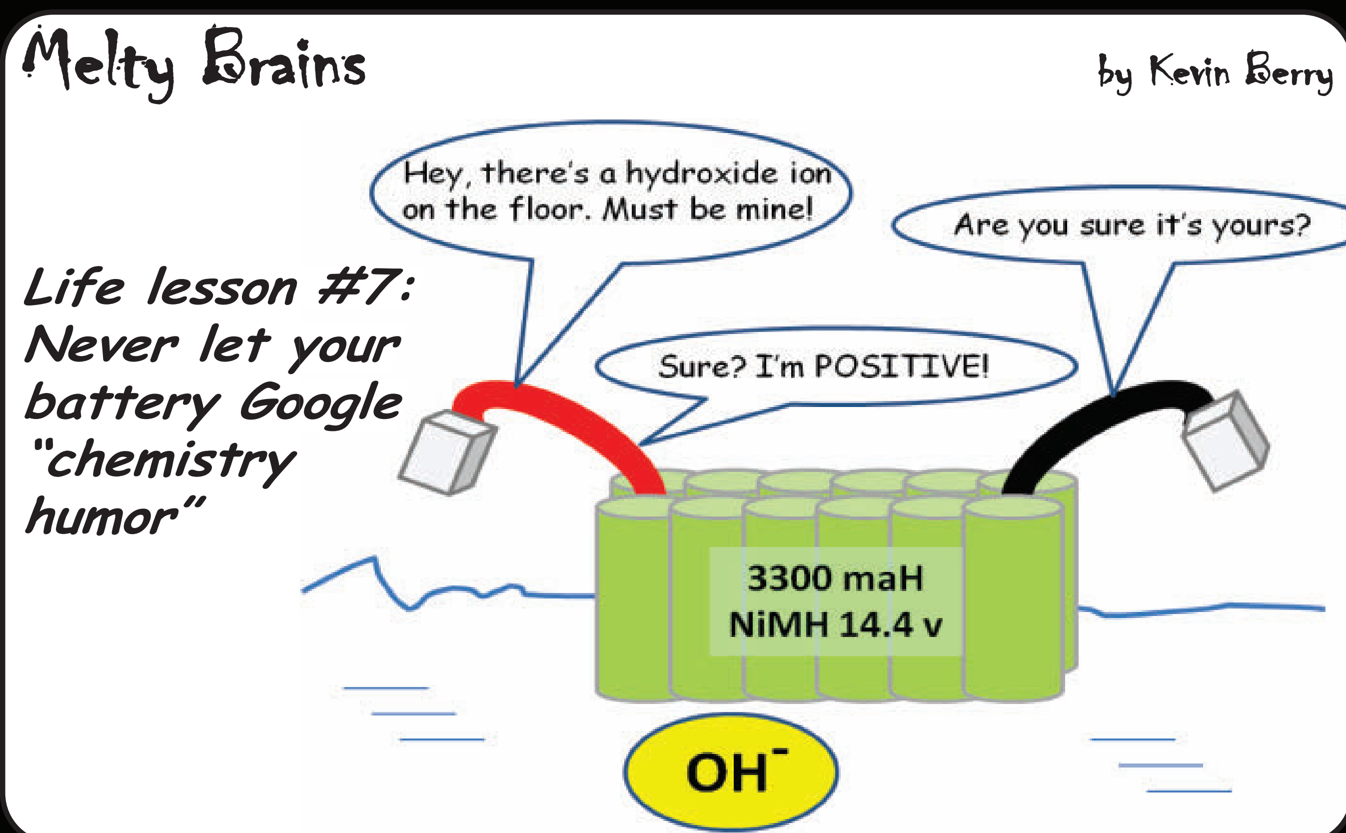
compartment. Plug it back into the socket it came out of and push the battery into the compartment (**Figure 7**). It was a snug fit but the cover still fits well.

I purchased new Futuba style chargers (**Figure 8**) from **www.towerhobbies.com** for around $17 each so that I could charge my new packs These plug straight in, and after a couple of hours of charging, the green LED on the radios (**Figure 9**) showed they had done their job. Well, three of them lit up green. The fourth refused to go beyond the yellow mid charge stage. That radio also seemed to discharge to yellow earlier than the rest when on alkalines, so I suspect the problem is with the charge level LED circuit on that radio and not with the battery or the charger. A day long event proved that the new batteries work as planned.

The cost per radio was about $36 but I've already spent at least half that per radio on alkalines this year. The ease of recharging and long life when charged means it's one less expense and worry for future events. **SV**


FIGURE 8.

FIGURE 9.



UPGRADING THE BOE-BOT

PART 2

by William Henning

In Part 1 of upgrading the Boe-Bot, I transformed a standard BASIC Stamp 2 based Boe-Bot to a fire breathing dragon — just kidding. I upgraded the whiskers to distance sensors, added wheel encoders to the wheels, and most important, replaced the Board of Education with an advanced multi-core 32-bit robot controller board: RoboProp. Robbie can now avoid ramming into walls and can run away if chased ... but it would be nice if he could do more than just randomly wander around the house (and irritate cats chasing him).

In this article, I will show you how to add:

- Infrared remote control so you can drive your robot around.
- Line following sensors so the robot can follow a track.
- A compass so you can tell what direction the robot is facing.
- An infrared range sensor so you can tell how far away the closest object is.

FIGURE 1. Robbie with all the features covered in this article.

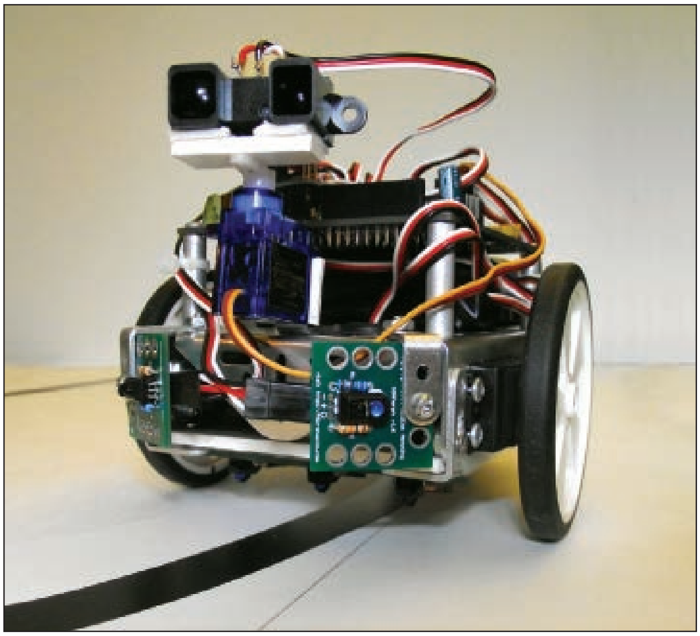


Figure 1 shows what Robbie looks like with the above additions.

INFRARED REMOTE CONTROL

One of the easiest and most useful additions to robots is the ability to receive infrared remote control commands. Note the universal remote shown in **Figure 2**. The original Boe-Bot kit includes two 38 kHz IR receivers.

Adding an IR remote sensor to RoboProp is very easy. As a matter of fact, you have a choice of making a movable sensor or permanently mounting the sensor on RoboProp.

To add a movable sensor, solder the IR sensor onto a small prototype board, and also solder the male pins of a three-pin servo extension cable to the appropriate legs. This way, you can just plug the cable into any of the servo/input three-pin headers. You can actually mount the IR sensor PCB anywhere on Robbie!

For a permanent sensor, solder the IR sensor into one of the two small prototyping areas on RoboProp along with the male pins of a three-pin servo extension cable. I suggest bending the sensor so its lens is pointing up at the ceiling. This way, it can receive IR signals that are bounced off the

LISTING 1. Simple RC driving mode code.

```
lspd := 0
rspd := 0

repeat

     robot.servo(lservo,1500+lspd)       ' lspd is a variable for left
                                         ' motor speed
     robot.servo(rservo,1500-rspd)       ' rspd is a variable for
                                         ' right motor speed

     cmd := robot.getir                  ' the RoboProp object makes
                                         ' it very easy to use Sony
                                         ' IR key codes

     case cmd
       "2":
            lspd += 25
            rspd += 25
       "4":
            lspd -= 25
            rspd += 25
       "6":
            lspd += 25
            rspd -= 25
       "8":
            lspd -= 25
            rspd -= 25
       "M": ' Mute button
            lspd := 0       ' setting motor speed to 0 stops the servo
            rspd := 0
```

FIGURE 2. Universal remote.

ceiling. I also mounted a compass on it.

Okay, now that the IR remote sensor is mounted, how do we make use of it? The most common IR codes are the 12-bit codes used by Sony remote controls. Fortunately, stores are full of inexpensive universal IR remotes. Universal remotes almost always have a large number of Sony code sets pre-programmed into them. I have been using cheap Universal 5-in-1 PX-RC1 remotes, and have had great success with the '000' Sony TV codes. **Table 1** shows the Sony codes returned by the SIRC object for the Propeller for Sony TV Type 000 (for the PX-RC1) for some common universal remote control buttons.

Now that we have a remote control with a known code set and an IR remote sensor on Robbie, it's time to write some code to drive Robbie around like a remote control truck. For my robots, I have standardized the following minimal driving controls:

- "2" — Increase forward speed.
- "8" — Decrease forward speed; decreasing after stopping makes robot go in reverse.
- "4" — Turn to left; multiple presses turn to left faster.
- "6" — Turn to right; multiple presses turn to right faster.
- "MUTE" — Emergency stop.

Listing 1 shows the main loop of the remote control driving mode.

You can find videos of Robbie being driven around by the IR remote control at **http://youtube.com/mikronauts**. You will find remote control driving code (and many more examples) at **http://Mikronauts.com/downloads**. At this point, Robbie can:

- Explore randomly without bumping into things.
- Run away when chased.
- Be driven around like a remote controlled robot.

However, I want Robbie to be capable of a lot more ...

LINE FOLLOWING

If you have watched YouTube videos or attended any robotics competitions, you have most likely seen robots following a black line on a white surface (or a white line on a dark surface). This is called "line following" and robots race against the clock to see who can complete navigating a course in the least amount of time.

You can follow lines with just one sensor, but this

Key	Code	Key	Code
1	$080	0	$089
2	$081	1-11 (returns "A")	$0E5
3	$082	2-12 (returns "B")	$09D
4	$083	< (vol +)	$092
5	$084	> (vol -)	$093
6	$085	+ (chan +)	$090
7	$086	- (chan -)	$091
8	$087	M (Mute)	$094
9	$088		

TABLE 1. Sony TV remote codes.

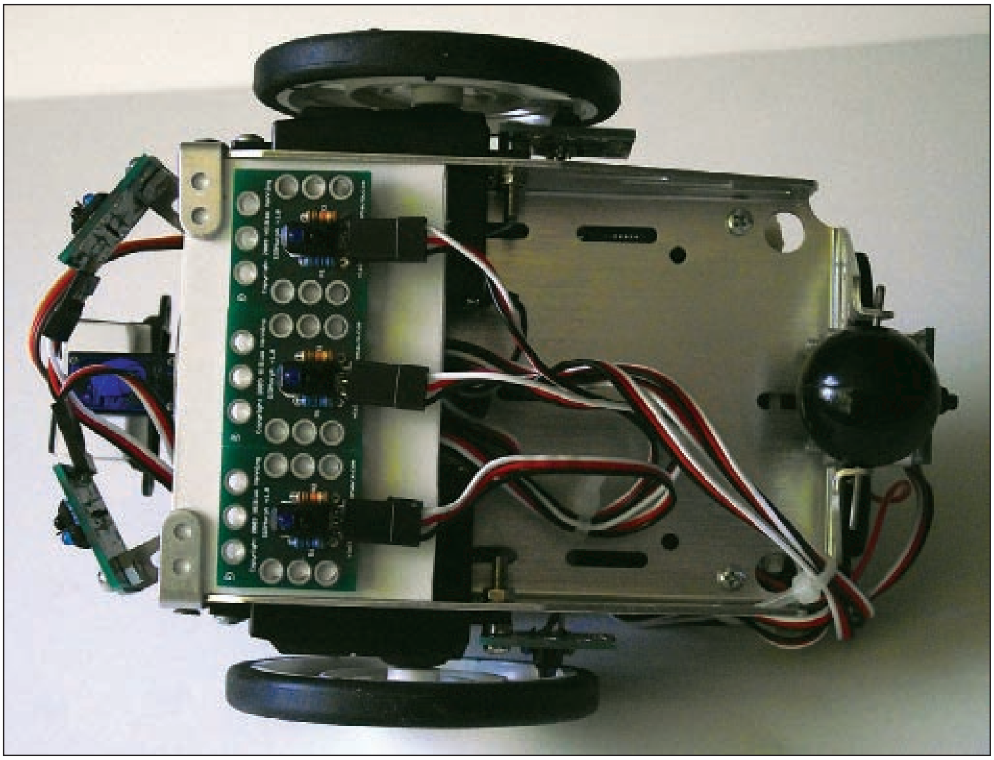


FIGURE 3. Line sensor mounted on Robbie.

as a five to seven (or more) sensor line following bar, using only three analog inputs.

There is a down side to using analog line detectors: You have to calibrate them, and they are influenced by the level of ambient lighting.

The good news is that by calibrating the sensors, you can handle dark tape on a light background or light tape on a dark background fairly easily. Please note that the better the contrast between background and line, the easier it will be for your robot to detect the line.

I like simplicity, so I use a very simple calibration scheme for line following:

- Take one reading with all three line sensors looking at a white background.
- Take one reading with all three line sensors looking at a line under each sensor on the background.

greatly limits the speed at which the robot can follow the line because it will have to hunt around more for the line to follow if it happens to leave the line. Two sensors work much better because the robot can keep going in the direction it is moving in until it "sees" that it is moving off the line by sensing the line on the left or right. Three sensors work even better. The robot can tell when the line is directly beneath the central sensor, and can also notice which direction it is drifting off the line when the line curves with even more sensors, the robot can tell how far it is off the line, and make more precise course corrections. However, it will use up more I/O pins and require more sophisticated programming.

For Robbie, I decided to build a line following sensor using three SirMorph IR sensor modules as shown in **Figure 3**.

SirMorph provides analog outputs from each sensor. This allows sensing how much of the line is in the field of view, thereby allowing more sophisticated software which can notice that the robot is leaving the central sensor and in which direction. This provides nearly the same capability

	Left Sensor	Center Sensor	Right Sensor	
Foam Core Board	>1000	>1000	>1000	White background
Black Tape	<650	<650	<650	Black tape
Reading 1	>1000	<650	>1000	Line @ center
Reading 2	<650	>1000	>1000	Line @ left
Reading 3	>1000	>1000	<650	Line @ right

TABLE 2. Test sample readings.

This allows RoboProp to save calibration values for each sensor for both "over line" and over "background" states.

The greater the difference between the over line and over background reading, the easier it will be for RoboProp to follow lines. If the difference is great enough, RoboProp will even be able to tell (roughly) where the line is relative to the three sensors. **Table 2** shows some test sample readings.

For illustration purposes, let's assume I have the background and line levels from **Table 2** when calibrating RoboProp. (I will supply some real values in a later table.)

You can see that for simple line following code, you can make judgments about where the line is simply by checking for (Sensor < 750) as that would clearly indicate which sensor the line was closest to.

Of course, you can detect if there is no line near any of the sensors by testing for all sensors having a value close to the background level; if your robot crossed a horizontal line during its travel, you would get a value close to the line level on all sensors.

Once you have the basic line following code working, you can experiment with controlling the rate of turning for the robot based on the "distance to line" value read from the sensor. That is, the closer a sensor reads to the line value, the closer that sensor is to the line. The converse is also true: The further a sensor is to the line, the closer the sensor value will be to the background level. This lets you know approximately where the robot is in relation to the line. The line sensors also allow robots to navigate a virtual maze.

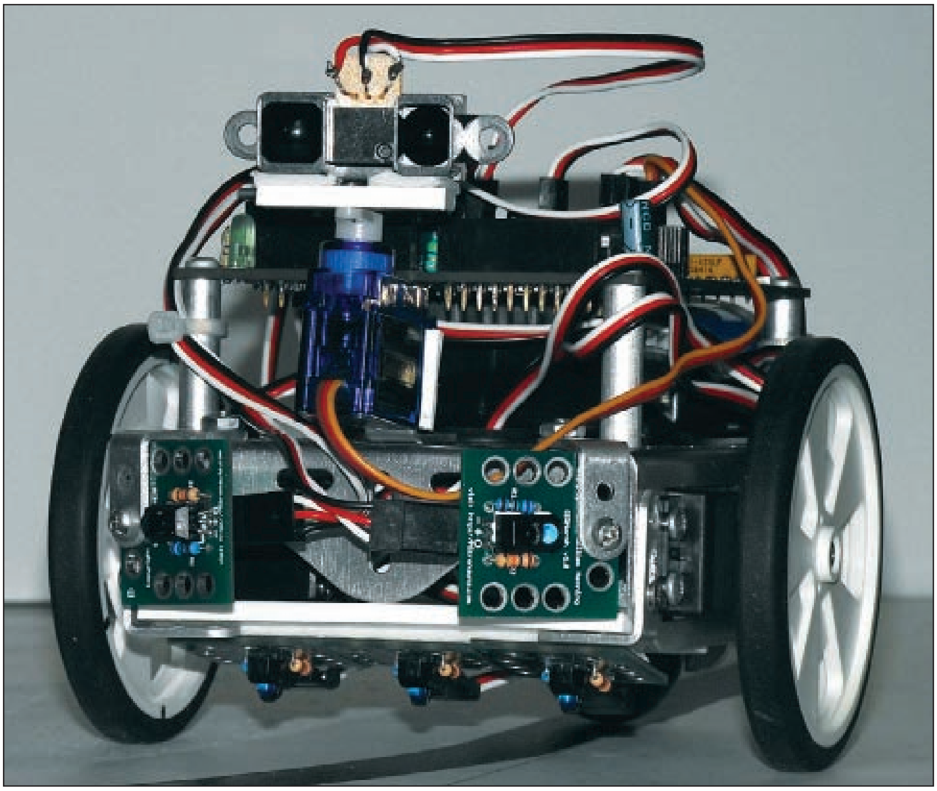


FIGURE 4. Robbie following a line.

You can either draw the walls with lines, or just draw a central single line wide path for the robot to follow.

At this point, Robbie can now also follow lines or avoid them. But I want Robbie to be capable of even more ... especially on his own!

ADDING A COMPASS

There are a lot of compass modules for sale these days, but after researching the integrated circuits used on them, I ended up choosing an HMC6352 module. This small, simple module is capable of self calibration and has one degree repeatability. Even more importantly, it communicates via a standard I^2C interface.

Parallax was out of stock when I wanted to order a module, but fortunately SparkFun had one based on the same chip.

Figure 5 shows the compass module on Robbie.

Now you know what the four-pin header was for. I added the small prototyping board with the IR remote sensor on it earlier, drawing power from the I^2C expansion header because I knew I would also be adding the compass module.

Since RoboProp has an I^2C expansion header, adding the compass was very simple:

- Wire up a small converter PCB as the I^2C pinout on RoboProp differs from the module.
- I added the Parallax HMC6352 object to the RoboProp library, and wrote Calibrate and GetHeading functions.

Initially, I got unexpected results — the heading jumped around too much. After doing some research, it turns out that there was a known bug in the Parallax HMC6352 object, and the suggested solution was to slow down the I^2C access rate. After a bit of sleuthing in the code, I found the problem — the code needed an additional delay between reading the two bytes of the response, so I added a small delay and voilà! No more corruption of the first bit in the second byte of the heading! (You can download the fixed HMC6352 code from **http://Mikronauts.com/downloads**.)

Now, why did I want to add a compass? There are many reasons!

- Know what direction the robot is facing in.
- Be able to turn to any compass heading.
- Be able to precisely turn left or right by the number of specified degrees.

If we combine the data from the compass and the wheel encoders, it becomes possible for the robot to navigate in its environment with fairly decent precision.

With the addition of a distance sensor, the robot can even be programmed to generate maps of its environment, and with further programming, it will be possible for a robot to recognize what room it is in and plan its route to different locations in different rooms, however, such mapping and route planning is beyond the scope of this article.

ADDING A RANGE SENSOR

Remember the front virtual bumpers and line sensors we added to Robbie? The SirMorph sensors are actually short range infrared distance sensors, but their range is too short for use as a "radar" style range/distance sensor. Currently, the most popular sensors used for robots for measuring distance to the nearest object are based on either infrared light or ultrasonic sound. One of the most popular ultrasonic sensors is the Parallax

```
repeat

  if  Line_Left
        right(50)
  if  Line_Right
        left(50)
  forward(50)
```

LISTING 2. Simple line following code using helper functions.

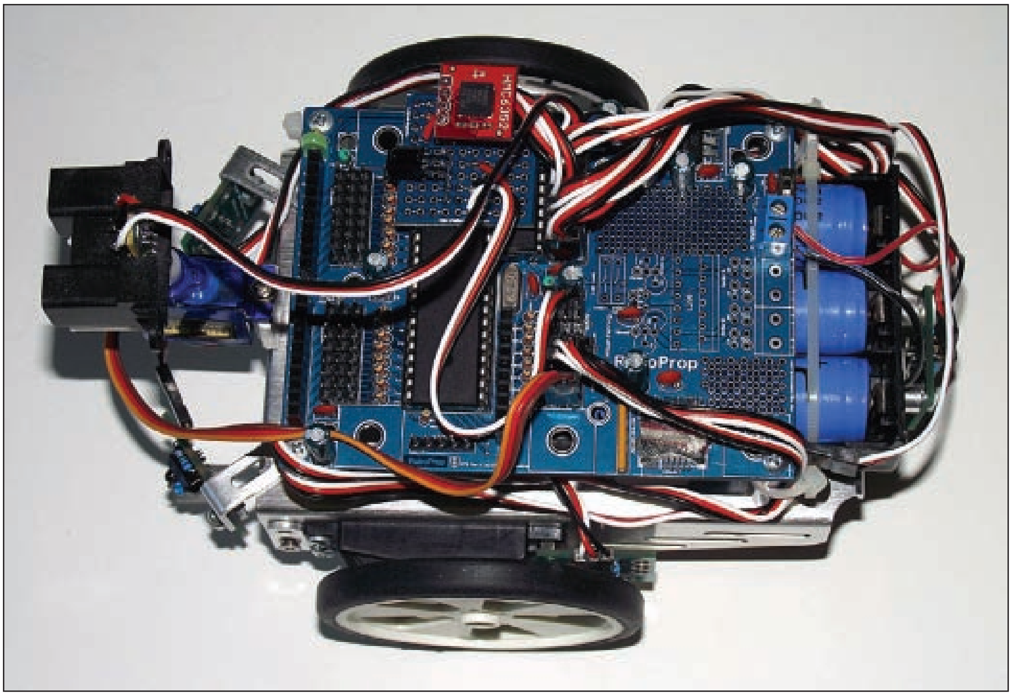


FIGURE 5. Compass module and IR remote sensor on Robbie.

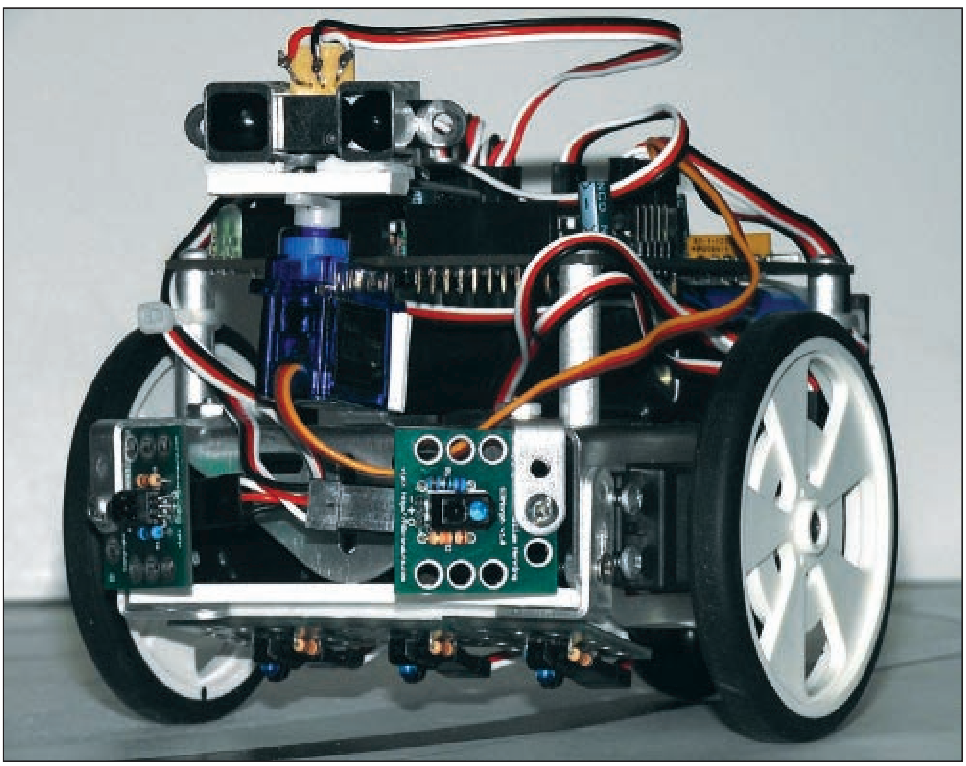
A/D converter on RoboProp to make room for the Sharp distance sensor's analog output.

I made a simple "pan" head for the Sharp range sensor from a TowerPro 9g micro servo, a Parallax "L" bracket mounted on the chassis with a 4-40 screw and nut, and two small pieces of 3 mm expanded PVC (Sintra).

Since I did not have a mating connector for the Sharp sensor handy, I just soldered a servo extension cable to the male pins of the connector after removing the plastic hood.

Ping))) sensor. You really can't go wrong with it, but I opted for a Sharp GP2Y0A02YK0F range sensor with a 0-5 VDC analog output and a 20 cm to 150 cm detection range (see **Figure 6**). Why? The Sharp sensor's narrower beam allows resolving objects with a smaller detection angle when they are not too close. Normally, you would not be using line sensors when you are using the front left/right and rear virtual bumpers or vice-versa; personally, I chose to disconnect the rear virtual bumper from the

FIGURE 6. Robbie with Sharp GP2Y0A02YK0F mounted on 9g TowerPro servo.



Let's say you want your robot to follow you around the house. One simple algorithm would be:

```
repeat
Scan 180 degrees from left to right, find the
closest object 30cm-150cm distant
Turn the robot until it faces the object
Go forward until the left or right virtual
bumper says you are too close, stop moving
```

What if you wanted to run away from the closest object?

```
repeat
Scan 180 degrees from left to right to find the
closest object 30-150cm
distant
Turn the robot until it
faces away from the closest
object
Go forward until there is
no object closer than 100cm
or a virtual bumper says
stop
```

A more sophisticated use of a scanning range sensor head is to generate a radar- or sonar-like

REFERENCES

38 kHz IR receiver TSOP34338
http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=751-1385-5-ND

Sony IR object sample code
http://obex.parallax.com/objects/477/

HMC6352 compass module
www.sparkfun.com/products/7915

Sharp GP2Y0A02YK0F range sensor
www.sparkfun.com/products/8958

```
' in the initialization code

robot.init_compass

' wherever you need to read the current heading just use

heading := robot.heading

' this assumes you declared the heading variable earlier
' as a WORD or a LONG
```

LISTING 3. Sample compass reading code.

plot of what is in front of the robot. I have been working on the default firmware for RoboProp, and I thought you might want to see the LIDAR plot generated by one of Robbie's cousins, Marco. Check out **Figure 7**.

ROBOPROP LIDAR DEMO

To capture this video, I added a small video output module to Marco and used a USB video capture device to record the live video coming from Marco. The heading is from an HMC6352 and was updating live — as was the radar plot showing the closest object in front of the Sharp sensor.

The rectangles under the radar plot show the status of the left/right/rear virtual bumpers, and the status of the three lines sensors on Marco. The other status displays are currently only mockups. (You can view the LIDAR demo on my YouTube channel at **www.youtube.com/mikronauts**.)

So, what am I planning to do with Robbie now?

- Explore more "reflex" behavior based on the virtual bumpers.
- Experiment with odometry for course/distance based navigation.
- Integrate a compass heading with the odometry based navigation.
- Add a small speaker so Robbie can make noises to indicate his status.
- Experiment with LIDAR based mapping.

I hope you enjoyed this article series. Please feel free to contact me at mikronauts@gmail.com with any questions you may have about Robbie (or RoboProp). **SV**

FIGURE 7. RoboProp LIDAR demo.

Double Your USB Pleasure With Cerebot

by Fred Eady

www.servomagazine.com/index.php?/magazine/article/october2011_Eady

What is better than a brand new Cerebot 32MX7? Two Cerebot 32MX7s! If you've followed my projects over the years, you know that it took quite a while for me to give a hug to USB. Despite the fact that I believe that RF engineers participate in pointy hat antics that relate to the zodiac, I have a fondness for RF projects. So, this month I'm going to shelve our Cerebot 32MX7 USB adventure in favor of a Cerebot 32MX7 802.15.4 project. Now you know why a pair of Cerebot 32MX7s are necessary. We've got some serious 32-bit RF embedded computing to do. So, let's get to it.

The MFB

You can think of the Digilent PmodRF2 as an MFB, or Magnetic Field Bender. The PmodRF2 is based on the Microchip MRF24J40 IEEE 802.15.4 2.4 GHz RF transceiver IC. Our little Digilent MFB is capable of altering magnetic fields to accommodate ZigBee and proprietary Microchip MiWi networking protocols. All we need to do to initiate the field bending is a PIC microcontroller that is capable of communicating via the SPI protocol. In that the Cerebot 32MX7 is based on the 32-bit PIC32MX795F512L, SPI portals are absolutely no problem.

The PmodRF2 hardware is configured to communicate as an SPI slave device with the SCK signal assuming a logically low idle state. Like all good data radio designs, the PmodRF2 includes a data available interrupt and a reset output.

ZigBee and MiWi data packets tend to be small in stature. So, the PmodRF2's ability to run at data rates of 250 kbps in IEEE mode and

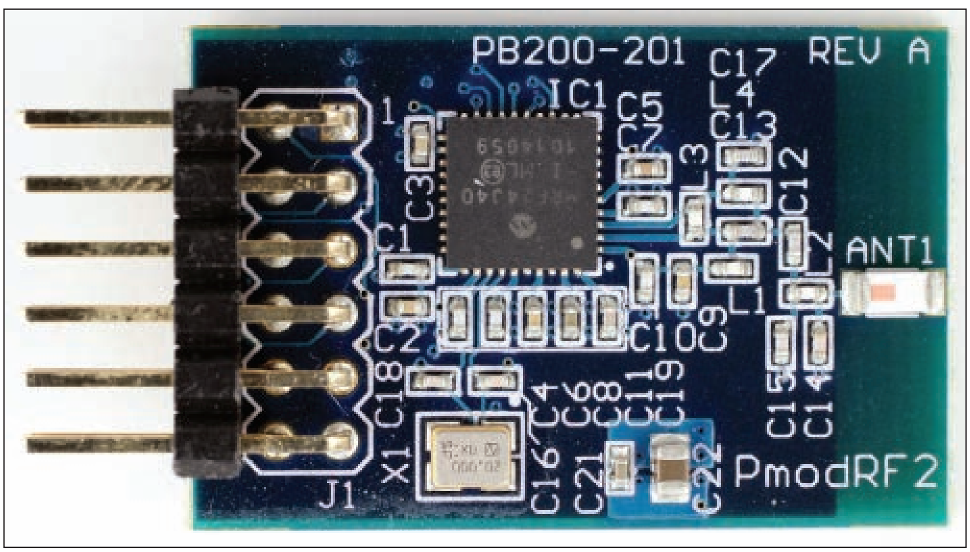


PHOTO 1. The PmodRF2 is based on the Microchip MRF24J40 2.4 GHz transceiver IC. Everything necessary to participate in an 802.15.4-based network including the antenna is soldered to the PmodRF2's copper-clad fiberglass printed circuit board.

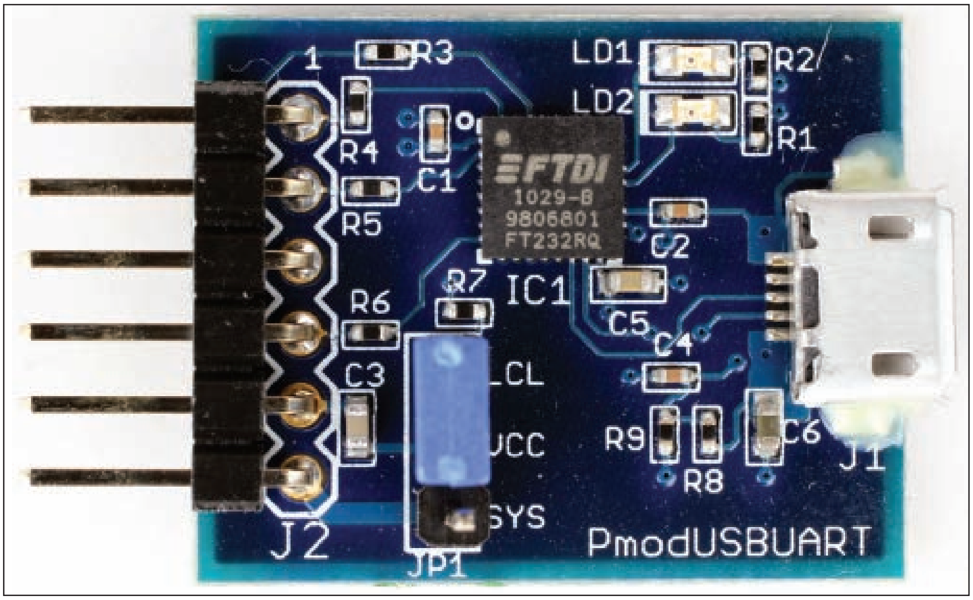


PHOTO 2. The PmodUSBUART is a direct replacement for the PmodRS232. This little bugger also works really good in solderless breadboard projects.

625 kbps in Turbo mode seems to be overkill. However, you really don't have to drive your Ferrari at 200 MPH either. In the case of the PmodRF2 and the Ferrari, the speed is there if we need it. A high speed PmodRF2 is stopped under the lens in **Photo 1**.

USB Wins Again

USB has been underbidding and winning contracts over RS-232 for some time now. Chalk up another win for the embedded USB interface. The PmodUSBUART positioned in **Photo 2** is based on the FTDI FT232RQ and will be used to bridge the data gap between the Cerebot 32MX7/PmodRF2 system and a PC terminal emulator.

The PmodUSBUART replaces the PmodRS232 module and the associated serial-to-USB cable. Pushing a power rail through an RS-232 connector was (and is) a pain in the ground plane. Like most other USB interface implementations, the PmodUSBUART module can be powered from the host system power supply or via the host system USB portal.

32-bit MiWi

As mentioned, MiWi is a Microchip protocol that can be used in very simple wireless 802.15.4 networks. You can get your own copy of MiWi by downloading the latest version of the Microchip Application Libraries. The majority of the applications found within the Application Libraries — including MiWi — are written for the out-of-the-box Microchip hardware development tools. Despite the various development tool configurations, the core device is a PIC. In the case of the Cerebot 32MX7, that microcontroller is the 32-bit PIC32MX795F512L. So, our job is to wedge the Cerebot 32MX7 hardware configuration into the existing MiWi hardware definitions. We'll use the Application Libraries' MiWi Simple Example for the PIC18 Explorer development board as our template.

Elbowing Into HardwareProfile.h

Wars begin with a single shot. The invasion of the MiWi Simple Demo's HardwareProfile.h file begins with this simple definition:

```
#define CEREBOT32MX7
```

The aforementioned *#define* declaration opens the door to a code space in which we will deposit our Cerebot 32MX7-specific hardware definitions.

The various PICs used in the Simple Demo application employ differing ways of implementing NVM (Non-Volatile Memory). Some microcontrollers use their internal EEPROM, while others may use external EEPROM or program Flash. We'll configure our Cerebot 32MX7 to use program Flash as its NVM element:

```
#if defined(CEREBOT32MX7)
     #define USE_PROGRAMMING_SPACE
```

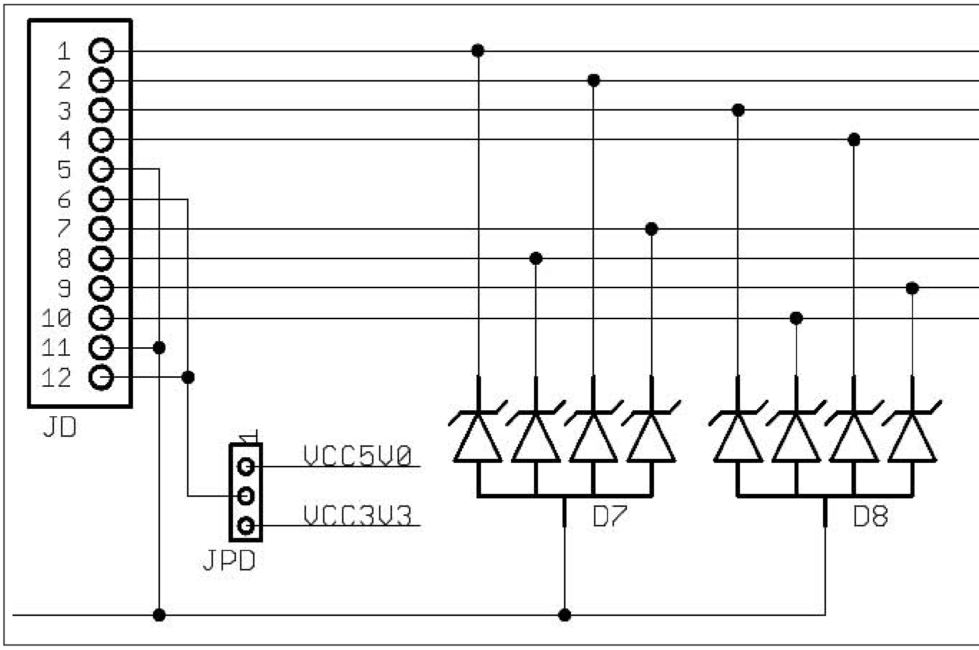
The Application Libraries applications must be written in such a way as to encompass all of the applicable PIC devices. For instance, as I mentioned earlier, we're going to base our modifications on the MiWi Simple Demo PIC18 Explorer code. The reason for this is that the MiWi code only supports the PIC32MX795F512L as a PIM mounted on the Explorer development board.

So, now that we know the modification base device, along the way we'll get rid of some of the obvious decisions embedded in the original code. Our first modification is a perfect example elimination of obvious code:

```
#if defined(PIC18_EXPLORER)
     #define CLOCK_FREQ           10000000
     #define USE_EXTERNAL_EEPROM
     #define EEPROM_SHARE_SPI

    #elif defined(MRF24J40)
          #define RFIF         INTCON3bits.INT1IF
          #define RFIE         INTCON3bits.INT1IE
```

We already know that the PmodRF2 is based on the MRF24J40. So, there's no need to base any coding decisions on any other radio. The Cerebot 32MX7's INT0, INT1, and INT2 external interrupt signal lines are not available to us to use in our ported MiWi application. So, we must do some work on the RFIF and RFIE definitions, which is basically take what we can get. The next available interrupt line is INT3. We've already taken care of the NVM space definition. That leaves only the CLOCK_FREQ



SCHEMATIC 1. The PmodRF2 will use the SPI portal services offered by the JD portal of the Cerebot 32MX7.

definition to be changed:

```
#define RFIF                    IFS0bits.INT3IF
#define RFIE                    IEC0bits.INT3IE
#define CLOCK_FREQ              64000000
```

Almost forgot. Nothing will work until we redefine the RF_INT_PIN as INT3 on the Cerebot 32MX7:

```
#define RF_INT_PIN              PORTAbits.RA14
#define RF_INT_TRIS             TRISAbits.TRISA14
```

The 32MX7 SPI portal that supports the PmodRF2 is located on the 32MX7 I/O portal JD. The PIC18 Explorer MiWi code's radio wake up hardware, chip select and reset signals don't match up with the I/O definitions of the 32MX7 I/O portal JD. So, we'll use some simple substitutions to keep as many of the PmodRF2 signals on the JD I/O pins as we can. Here are the original PIC18 Explorer definitions:

```
#define PHY_CS                  LATBbits.LATB3
#define PHY_CS_TRIS             TRISBbits.TRISB3
#define PHY_RESETn              LATBbits.LATB5
#define PHY_RESETn_TRIS         TRISBbits.TRISB5
#define PHY_WAKE                LATBbits.LATB4
#define PHY_WAKE_TRIS           TRISBbits.TRISB4
```

These are our 32MX7-adjusted I/O definitions:

```
#define PHY_CS                  LATDbits.LATD9
#define PHY_CS_TRIS             TRISDbits.TRISD9
#define PHY_RESETn              LATDbits.LATD2
#define PHY_RESETn_TRIS         TRISDbits.TRISD2
#define PHY_WAKE                LATDbits.LATD3
#define PHY_WAKE_TRIS           TRISDbits.TRISD3
```

A quick peek at **Schematic 1** verifies that the PmodRF2's chip select and reset signals can be found on the Cerebot 32MX7's JD I/O portal. **Schematic 1** also informs us that the SPI portal signals are part of JD I/O portal. The PIC18 Explorer wants to see the SPI signals on these pins:

```
#define SPI_SDI                 PORTCbits.RC4
#define SDI_TRIS                TRISCbits.TRISC4
#define SPI_SDO                 LATCbits.LATC5
#define SDO_TRIS                TRISCbits.TRISC5
#define SPI_SCK                 LATCbits.LATC3
#define SCK_TRIS                TRISCbits.TRISC3
```

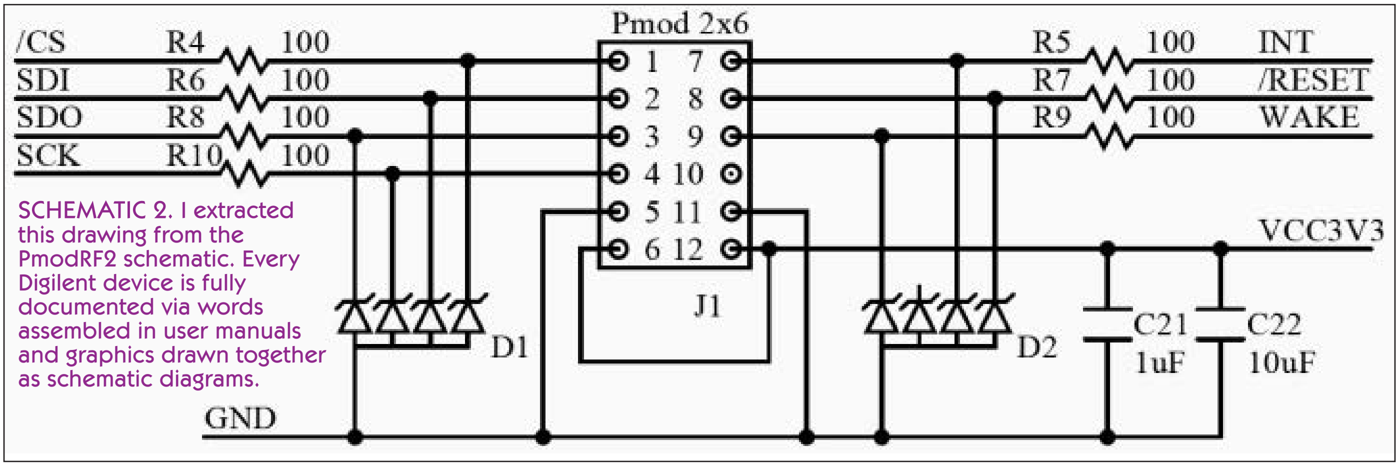
Schematic 1 says we need to place the 32MX7 SPI portal signals under these pins:

```
#define SPI_SDI                 PORTCbits.RC4
#define SDI_TRIS                TRISCbits.TRISC4
#define SPI_SDO                 LATDbits.LATD0
#define SDO_TRIS                TRISDbits.TRISD0
#define SPI_SCK                 LATDbits.LATD10
#define SCK_TRIS                TRISDbits.TRISD10
```

At this point, we've logically attached the PmodRF2 to the 32MX7. And, as you have witnessed, it wasn't rocket science. The physical attachment of the PmodRF2 is just as straightforward and doesn't require any additional astronaut or cosmonaut training on your part. The reason for this is that the "D" in Digilent stands for documentation. There isn't any pin of any Digilent device that is undocumented. To prove my point, you can link up our logical Cerebot 32MX7-to-PmodRF2 connections by cross referencing **Schematics 1** and **2** which are both excerpts from Digilent documents.

In addition to superior documentation, Digilent is also not afraid to provide the necessary wire and connectors to make things happen. All of the cables and headers you need to convert logical connections to physical connections are included with your purchase of the PmodRF2. Even with Digilent's documentation diligence, it never hurts to place a picture behind the words. The physical PmodRF2 connections are captured in **Photo 3**.

With the PmodRF2 "plugged in," we can turn our attention to the things that really matter — buttons and LEDs:



SCHEMATIC 2. I extracted this drawing from the PmodRF2 schematic. Every Digilent device is fully documented via words assembled in user manuals and graphics drawn together as schematic diagrams.

```
#define PUSH_BUTTON_1          PORTDbits.RD13
#define PUSH_BUTTON_2          PORTGbits.RG7
#define LED_1                  LATGbits.LATG12
#define LED_2                  LATGbits.LATG13
#define LED_3                  LATGbits.LATG14
#define LED_4                  LATGbits.LATG15
#define BUTTON_1_TRIS          TRISDbits.TRISD13
#define BUTTON_2_TRIS          TRISGbits.TRISG7
#define LED_1_TRIS             TRISGbits.TRISG12
#define LED_2_TRIS             TRISGbits.TRISG13
#define LED_3_TRIS             TRISGbits.TRISG14
#define LED_4_TRIS             TRISGbits.TRISG15
#define TMRL TMR2
```

Once again, no Air Force training is required and believe me, these 32MX7 LED and button I/O positions are well documented. Note that we also twiddled with the 32MX7's TMRL definition.

Intrusion of Hardware Profile.c

Rather than do a blow-by-blow description of the MiWi Simple Demo HardwareProfile.c modifications, I've included a copy of the modified HardwareProfile.c code snippet in the download package that you can reference as I describe what the code is designed to do. The 32MX7 code snippet is inserted into the original Simple Demo HardwareProfile.c file just prior to the *#elif defined(PIC18_EXPLORER)* statement. In fact, all of our 32MX7 modifications fall prior to the original PIC18 Explorer code whenever possible.

The first order of business is to configure and activate the Cerebot 32MX7's SPI1 portal. The PmodRF2 is an SPI slave device. So, we must configure the 32MX7's SPI1 portal to operate in SPI Master mode with the idle SPI clock level defined as logically low. The SPI1CON bits are set to force data transfer on the falling edge of the SPI clock signal. Once the SPI portal is configured and activated, the speed of the SPI portal must be regulated in accordance with the PIC32MX795F512L's clock frequency. Following the establishment of a suitable SPI data exchange speed, the characteristics of the external interrupt (INT3) are configured.

The final program actions aimed at arming INT3 are performed in the MRF24J40.c file:

```
#elif defined(__PIC32MX__)
       #if defined(CEREBOT32MX7)
             void __ISR(_EXTERNAL_3_VECTOR, ipl4)
             _INT1Interrupt(void)   //mx7
       else
```

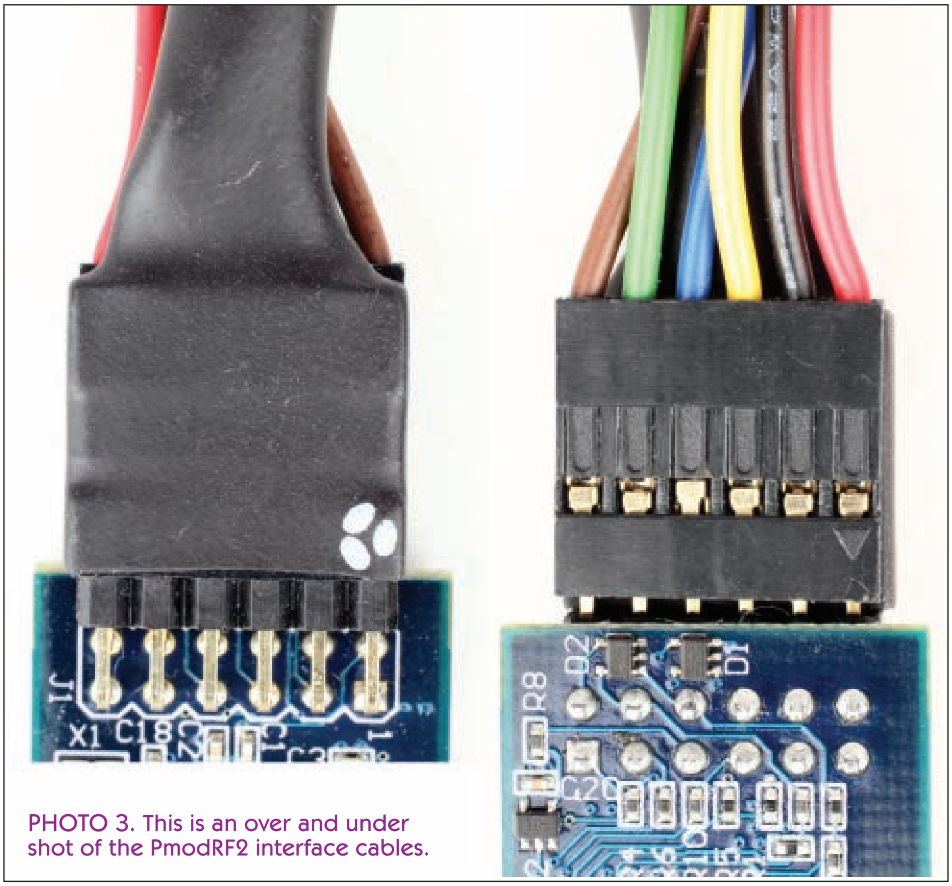
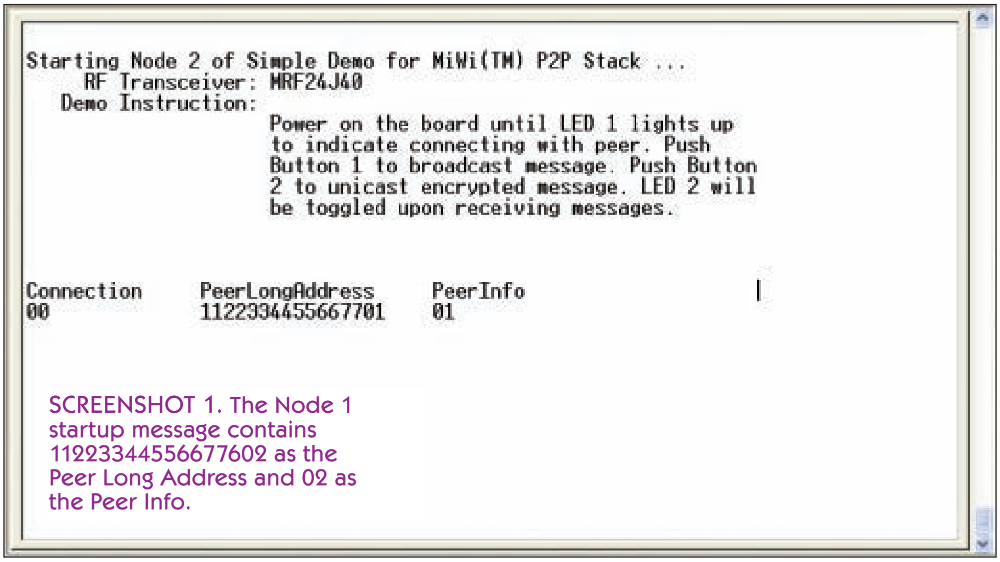


PHOTO 3. This is an over and under shot of the PmodRF2 interface cables.



SCREENSHOT 1. The Node 1 startup message contains 1122334455667602 as the Peer Long Address and 02 as the Peer Info.

Sources

Digilent
Cerebot 32MX7
PmodRF2
PmodUSBUART
www.digilentinc.com

Microchip
Microchip Application Libraries
PIC18 Explorer
www.microchip.com

Fred Eady can be reached via email at fred@edtp.com.

```
        void __ISR(_EXTERNAL_1_VECTOR, ipl4)
        _INT1Interrupt(void)//mx4
    #endif
    #else
        void _ISRFAST _INT1Interrupt(void)
```

We are ready to attempt a compile. If the compile flies, we'll make an approach at loading the code via the 32MX7's Area 51 programmer/debugger complex.

Just Because It Compiles ...

Doesn't mean it will work. And in this case, it doesn't. We've got a bit more work to do. The good news is that at least the 32MX7-to-32MX7 RF connection is working. Both Cerebot 32MX7s are using their respective PmodUSBUARTs to post startup messages via the terminal emulators. **Screenshot 1** is a capture of the Node 2 startup message. I have an almost identical set of information on the Node 1 terminal emulator session.

I do read instructions every now and then. LED1 does indeed illuminate on both nodes. However, the pushbuttons are dead. So, the trail starts at the pushbutton definitions for both the PIC18 Explorer and the 32MX7. The pushbutton code for both development platforms lies within the confines of the HardwareProfile.h file. Here are the PIC18 Explorer assignments:

```
#define PUSH_BUTTON_1      PORTBbits.RB0
#define BUTTON_1_TRIS      TRISBbits.TRISB0
#define PUSH_BUTTON_2      PORTAbits.RA5
#define BUTTON_2_TRIS      TRISAbits.TRISA5
```

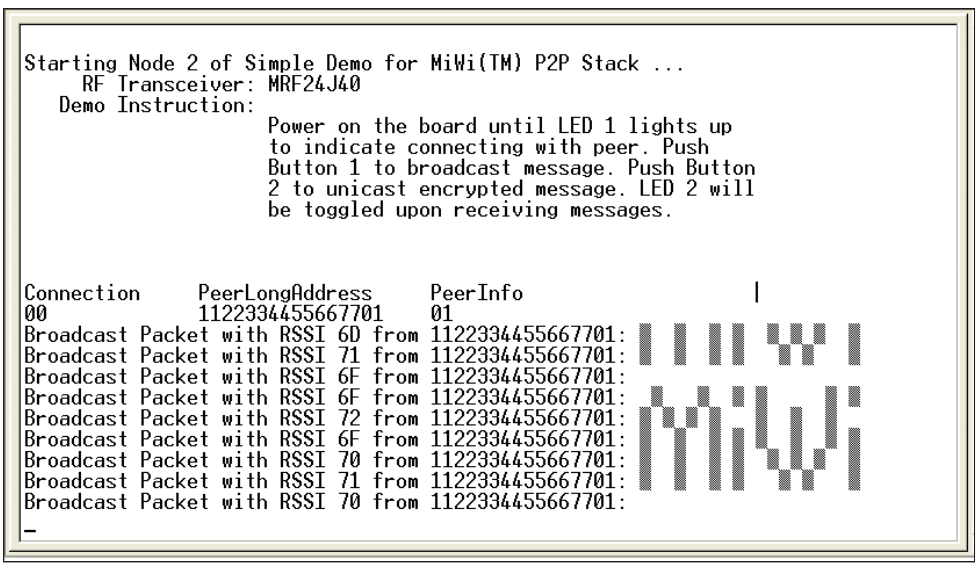
I went to the Microchip website and dialed in the PIC18 Explorer schematic. I pulled the pushbutton portion of the PIC18 Explorer schematic into **Schematic 3**. Now, let's pull the 32MX7 pushbutton schematic snippet into **Schematic 4**. Do you see what we need to fix?

The next step we need to take is to find the pushbutton code in the MiWi Simple Demo application. The pushbutton function is not located in the main application file but is found in the HardwareProfile.c file:

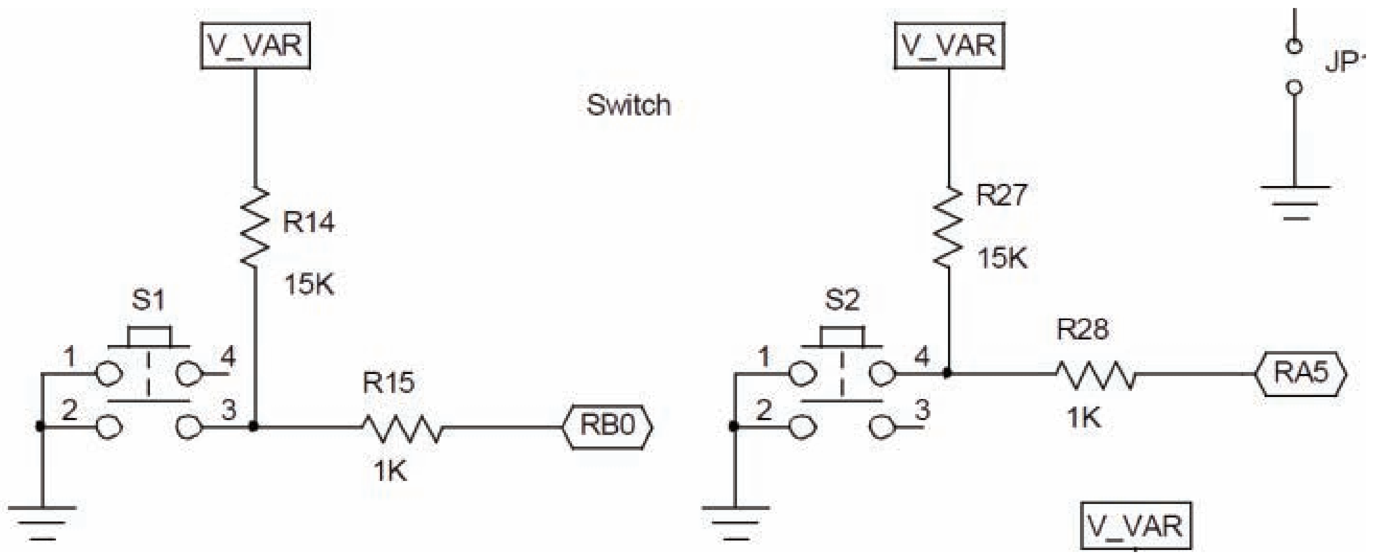
```
MIWI_TICK tickDifference;

if(PUSH_BUTTON_1 == 0)
{
    #ifdef ENABLE_SLEEP
        while(PUSH_BUTTON_1 ==
        1 );
        return 1;

    #else
     //if the button was
     //previously not pressed
     if(PUSH_BUTTON_pressed
     == FALSE)
     {
         PUSH_BUTTON_pressed
         = TRUE;
         PUSH_BUTTON_press_
         time = TickGet();
         return 1;
```

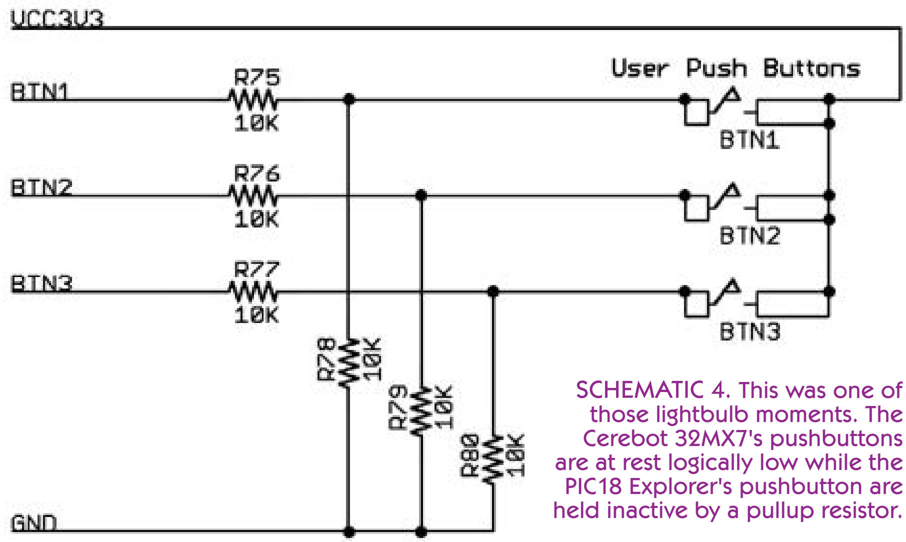


SCREENSHOT 2. Once the pushbutton logic snafu was squelched, data flowed in abundance on the PmodRF2 link.



SCHEMATIC 3. This is so obvious. It took me the better part of an afternoon to discover the PIC18 Explorer's pushbutton logic level inversion.

```
    }
  #endif
}
else if(PUSH_BUTTON_2 == 0)
{
  #ifdef ENABLE_SLEEP
    while(PUSH_BUTTON_2
    == 1);
    return 2;
  #else
    //if the button was
    //previously not pressed
    if(PUSH_BUTTON_pressed
    == FALSE)
    {
      PUSH_BUTTON_pressed
       = TRUE;
      PUSH_BUTTON_press_
      me = TickGet();
      return 2;
    }
  #endif
}
```



SCHEMATIC 4. This was one of those lightbulb moments. The Cerebot 32MX7's pushbuttons are at rest logically low while the PIC18 Explorer's pushbutton are held inactive by a pullup resistor.

The pushbutton code was snatched from the Node 2 framework. The code works for the PIC18 Explorer as the depressed state of the pushbuttons is logically low. We need only make this simple logic inversion for the 32MX7 pushbuttons to be recognized:

```
if(PUSH_BUTTON_1 == 1)

else if(PUSH_BUTTON_2 == 1)
```

Naturally, a similar code change must be made on the Node 1 side.

A couple of compiles and reloads later, LED2 toggled on the receiving 32MX7 with the depression of each 32MX7 transmitter pushbutton. According to the comments in the code, Node 1 should transmit a character map with each pushbutton 1 depression. Here's the character map:

```
ROM const BYTE MiWi[5][21] =
{
{0x20,0xB2,0x20,0x20,0x20,0xB2,0x20,0x20,0xB2,0x2
0,0xB2,0x20,0x20,0x20,0x20,0x20,0xB2,0x20,0xB2,0x
0D,0x0A},
{0xB2,0x20,0xB2,0x20,0xB2,0x20,0xB2,0x20,0x20,0x2
0,0xB2,0x20,0x20,0xB2,0x20,0x20,0xB2,0x20,0x20,0x
0D,0x0A},
{0xB2,0x20,0x20,0xB2,0x20,0x20,0xB2,0x20,0xB2,0x2
0,0xB2,0x20,0x20,0xB2,0x20,0x20,0xB2,0x20,0xB2,0x
0D,0x0A},
{0xB2,0x20,0x20,0xB2,0x20,0x20,0xB2,0x20,0xB2,0x2
0,0x20,0xB2,0x20,0xB2,0x20,0xB2,0x20,0x20,0xB2,0x
0D,0x0A},
{0xB2,0x20,0x20,0xB2,0x20,0x20,0xB2,0x20,0xB2,0x2
0,0x20,0x20,0xB2,0x20,0xB2,0x20,0x20,0x20,0xB2,0x
0D,0x0A}
};
```

If you associate 0x20 as a space character and 0xB2 as a darkened block, you can render the message on a sheet of graph paper. However, it's a bit easier to just press the button and capture the MiWi graphic at Node 2 as I did in **Screenshot 2**. The RSSI (Received Signal Strength Indicator)

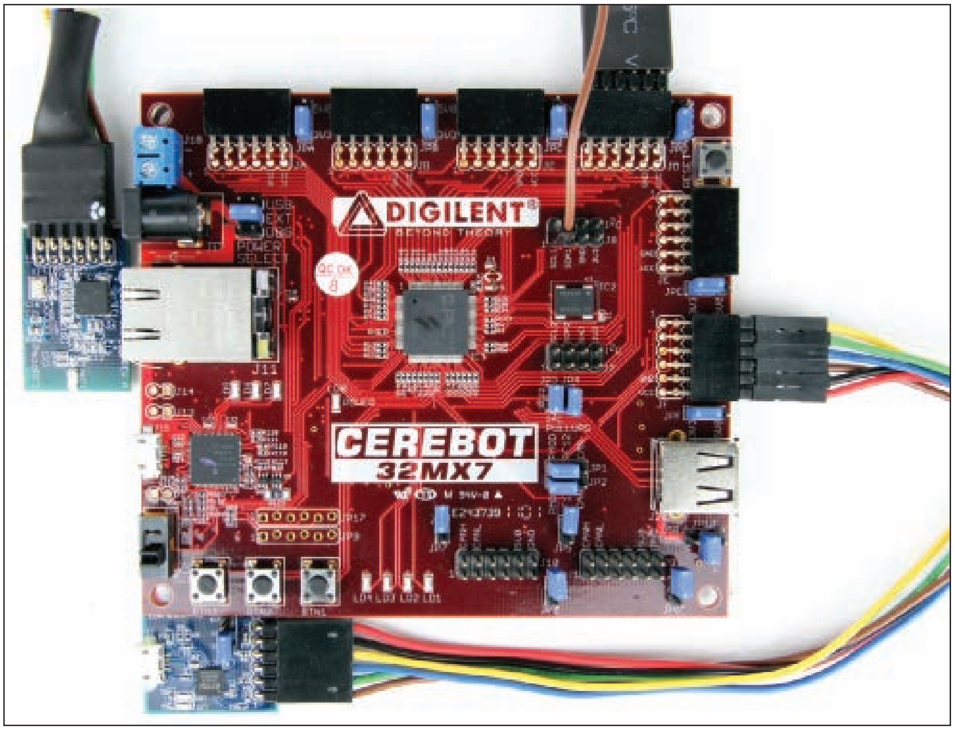


PHOTO 4. Here is a family portrait that includes the PmodRF2, the PmodUSBUART, the Cerebot 32MX7, and all of the aunt cables and uncle headers.

is generated by the receiving node (Node 2).

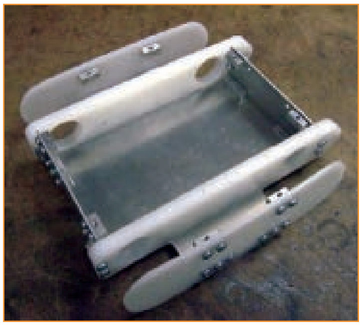
Kids, Please Try This At Home

All of the physical components for a single node are collected in **Photo 4**. I am including the entire MPLAB project in the download package. You will need to load up a copy of the v2010-10-19 version of the Application Libraries to run this month's application right out of the box.

The MiWi Simple Demo code we ported includes everything you need as a template to write your own custom 32MX7/PmodRF2 P2P applications. The MiWi/P2P RF connection mechanism is there. The MiWi transmit and receive data handling routines are there. And, there is a MiWi light switch application hidden in the bushes, too. **SV**

Continued from page 21

wedge or brick design, it features a combination of precision watercut 7075 aluminium and template routed UHMW panels.



Designed around four of the popular 1,000 RPM motors, it provides enough internal space for the required batteries and speed controllers, a solid mounting location for a 0.5" wedge pivot pin, and solid construction combined with easy access for repairs.

This kit is perfect for the beginner in combat robots as it combines a simple-to-build design with excellent performance and great survivability.

Options included fitted motors all the way to turnkey solutions. Estimated street prices begin at $120.

Gearmotor Mounting Brackets

Kitbots also introduces new mounting brackets for the popular FingerTech Spark or Silver Spark gearmotors.

The plates are precision watercut from 0.030" 7075 aluminum and come with the all required mounting screws. Parts provided are sufficient to mount two gearmotors.

Weight per motor is approximately 0.05 oz. Estimated street price is $20 a set.

For further information on the previous two new products, please contact:

Kitbots	Website: **www.kitbots.com**

Show Us What You've Got!

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

Designing A Low Cost Laser – Part 1

by Joe Grand

www.servomagazine.com/index.php?/magazine/article/october2011_Grand

In early 2008, Ken Gracey from Parallax approached me about designing a low cost Laser Range Finder (LRF) module. Sure, there were (and still are) lots of other sensing options available such as infrared, ultrasonic, accelerometers, gyroscopes, and magnetic, and people have hacked off-the-shelf sensing products like Nintendo's WiiMote and Microsoft's Kinect, but no sensor was available to the hobbyist in a compact, easy-to-interface module that used laser light to determine the distance between itself and a target object. I liked the idea and took on the challenge.

Searching for the Light

This article shares the highlights of my development journey and details the features of the final product (Parallax #28044). All engineering documentation and resources are available on my website at **www.grandideastudio.com/portfolio/laser-range-finder**/. The LRF is released under a Creative Commons Attribution 3.0 United States license (**http://creativecommons.org/licenses/by/3.0/us**/), allowing folks to hack on and modify the design as they wish.

Early Attempts

I originally had started developing a system using the time-of-flight method (**http://en.wikipedia.org/wiki/Time-of-flight** and **www.repairfaq.org/sam/laserlia.htm#liarfbl**) which measures the time required for a laser pulse to travel to the target object and back. Knowing the speed of light and the travel time, a simple calculation is all that's necessary to determine distance. Conceptually, time-of-flight is easy to understand and explain, but not easy to achieve in practice as complex, high speed transmission and detection systems are required.

I built a high speed time-to-digital converter using the ACAM GP2 (**www.acam-usa.com/GP2.html**) and a Parallax SX that would be used to measure the elapsed time of the laser travel. I also put together a programmable pulse generator using the Data Delay Devices 3D3608 (**www.datadelay.com/asp/oscpg.asp**) which would generate pulses ranging from 14 ns to 1.3 µS that I thought I could use to trigger my not-yet-designed laser driver circuitry.

Being primarily an embedded systems/digital designer, the circuitry required for the system was far too deep in the analog domain for me, so I scrapped the approach in hopes of finding a method more suitable to my skill set.

Optical Triangulation

I eventually decided to go with the method of optical triangulation, where the distance to a targeted object is calculated using triangulation with simple trigonometry

Range Finder

between the center points of laser light, camera, and object. The most compelling example is the webcam-based DIY laser range finder (**http://sites.google.com/site/todddanko/home/webcam_laser_ranger**); my design is based — in theory — on this implementation.

Referring to **Figure 1**, a laser diode module shines a laser spot onto the target object. The value *h* is a fixed, known distance between the center points of the laser diode and the camera. When the distance to the target object *D* changes, so do both the angle q and the value *pfc (pixels from center)* which is the number of pixels the centroid of the primary blob (laser spot) is away from the camera's center point. As the object gets closer, the value of *pfc* (and angle q) increases. As the object gets farther away, *pfc* (and angle q) approaches zero. Parallax's Beau Schwabe made a very short video that demonstrates this phenomenon: **http://forums.parallax.com/showthread.php?89395-Laser-stuff-for-those-interested&p=613631**.

If we know the angle q, then we can use basic trigonometry to calculate the distance value D:

$$\tan q = h / D$$

Solving for D, we get:

$$D = h / \tan q$$

Since the camera system returns a

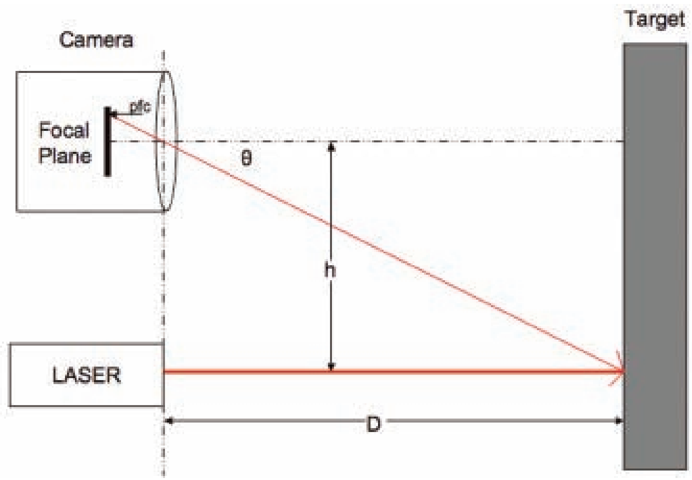


FIGURE 1. *Theory of operation for a webcam-based DIY laser range finder.*

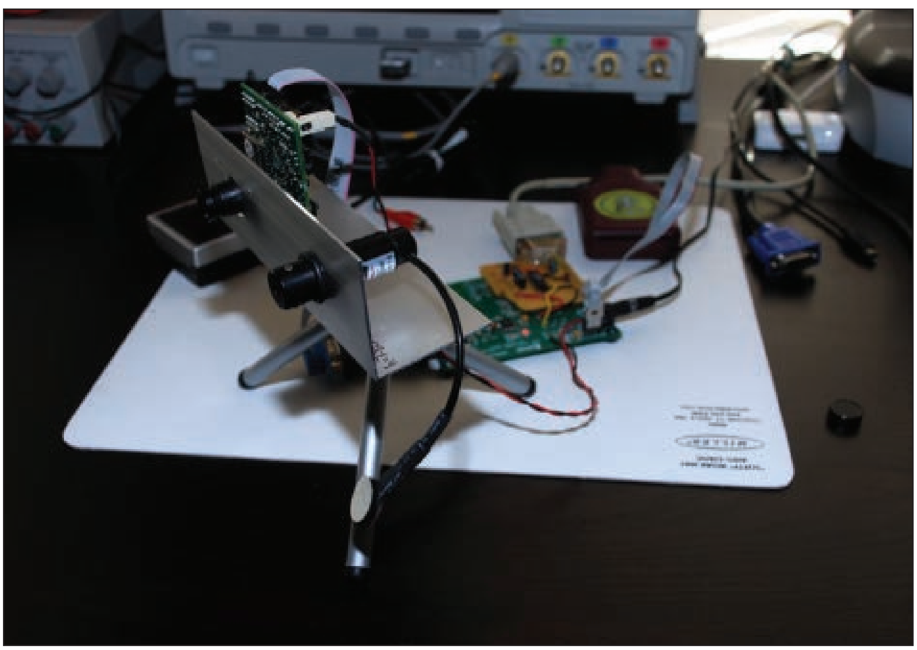


FIGURE 2. *My first successful laser range finder prototype. The CMUcam2 and manually-controlled laser diode are mounted to an aluminum bracket held by a tripod. The control circuitry is located on the PCB in the background.*

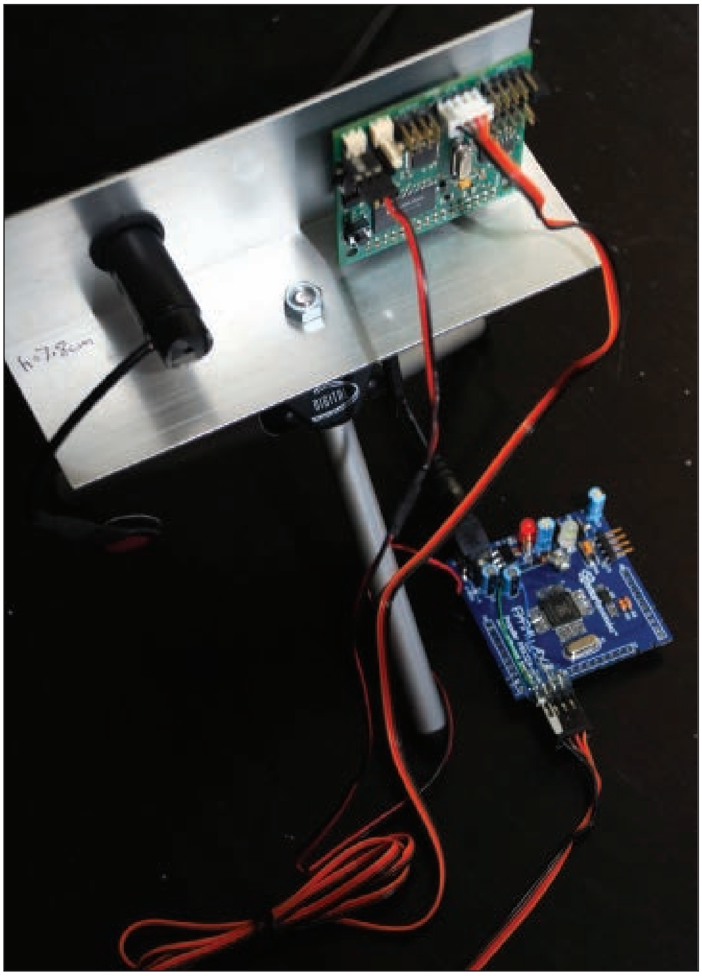


FIGURE 3. *The second LRF prototype. This time, I used a Propeller to handle the CMUcam2 communication and range finding math*

pfc value, we need a way to correlate the value with angle q. The relationship between the *pfc* and angle can be described with a slope-intercept linear equation (**www.math.com/school/subject2/lessons/S2U4L2GL.html**).

So, once we shine the laser onto the target object and receive the *pfc* value of the laser spot, an angle q can be calculated using the slope-intercept equation and passed to the trigonometric function to determine the actual distance the range finder is from the target object. (Phew!)

Proving the Concept

I selected the Omnivision OVM7690 640x480 CMOS camera module (**www.ovt.com/products/sensor.php?id=45**) and the Parallax Propeller to be the primary components of my final LRF system. I built a series of prototypes to prove the concept of optical triangulation and to ensure that I could generate measurements that were practical and usable within a portable, low cost module.

First, I created a system using a CMUcam2 (**www.cmucam.org**) and a Freescale QG8 microcontroller (see **Figure 2**). I've had extensive experience working with Freescale parts and had some good baseline code to start with. The prototype worked well given the low CMUcam2 resolution (176x255) and had a measurement range of seven inches to 40 inches. (Check out the video I made to demonstrate the system at **www.youtube.com/watch?v=-h5ctq7dE9k**.)

Next, I transitioned from the Freescale QG8 to the Propeller (see **Figure 3**). It wasn't too difficult to create my initial framework, communicate with the CMUcam2 through a serial interface, and handle the range finding math using some floating point libraries.

Both prototypes confirmed that I was on the right path with optical triangulation, so it was time to continue development with an actual OVM7690 camera and create my own image processing routines (instead of relying on those provided by the CMUcam2). I put together a custom development "platform" using a Propeller proto board, OVM7690, and interface circuitry. I used this setup exclusively until finally moving to the true-to-form PCB (printed circuit board) later in the process (see **Figure 4**).

Laser Diode Selection

Tangential to building my proof-of-concept prototypes, I spent some time researching and evaluating various laser diode/driver solutions to provide the red laser spot used for optical triangulation. I identified three options, each which had advantages and disadvantages:

1). **Discrete APC (Automatic Power Control)**. There are dozens of schematics and solutions online for driving CW (continuous wave) laser diodes. If you've ever taken apart a cheap laser pointer, there's usually a discrete driver solution. While you can get by driving a laser diode with a single current limiting resistor, you'll likely damage the laser diode with any change in drive voltage or transient spike. A proper discrete circuit requires not only driving the laser diode, but also watching the laser's monitor photodiode and continually adjusting the drive output power. Sam's

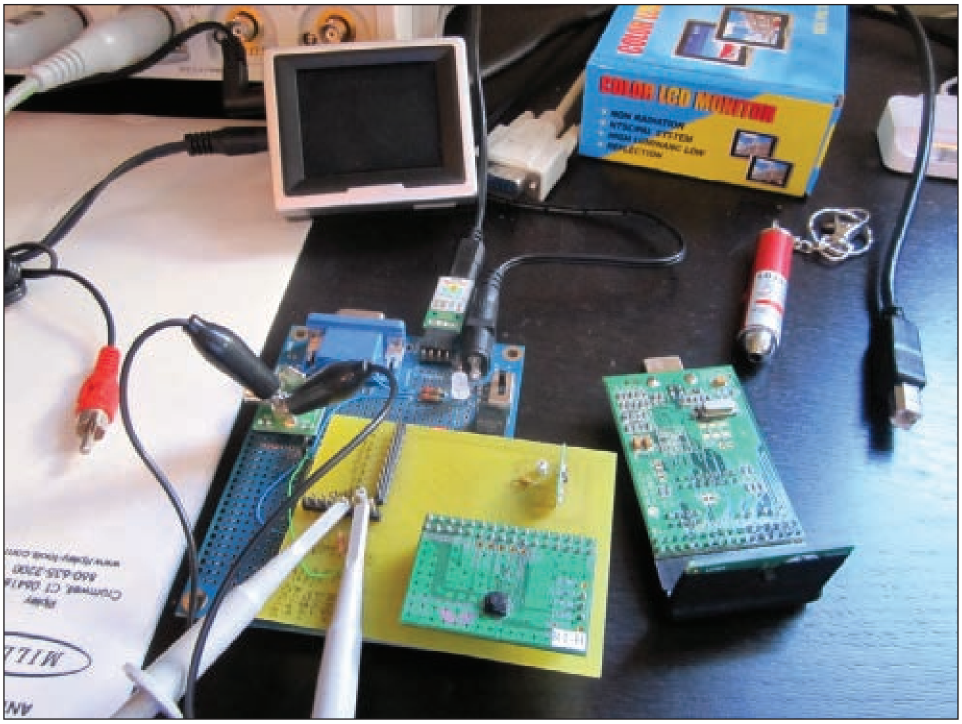


FIGURE 4. *I used this custom development platform for the majority of the project.*

Laser FAQ (**www.repairfaq.org/sam/laserdps.htm#dpstoc**) has a bunch of schematics of varying complexity. The advantage to this solution is price — you can get by with a handful of parts for or under $1 in quantity. However, creating a laser driver with a discrete APC solution that actually works reliably in production capacity can be a challenge.

2). **Integrated circuit (IC) driver**. An IC driver contains the necessary drive circuitry, APC, and laser diode protection mechanisms in a single package, though an external laser diode and collimating lens are still required. A very attractive part is the iC-WK universal laser driver from iC-Haus (**www.ichaus.de/product/iC-WK%20iC-WKL**). The iC-WK comes in an eight-pin SOIC or MSOP package, provides support for all three common laser diode configurations (N-type, P-Type, and M-type), and provides CW drive current up to 90 mA. Ballpark pricing for the iC-WK is ~$2.70 for 1K pieces. An evaluation board was readily available which allowed me to get up and running in minutes (all I needed to do was adjust the maximum allowable output power using a single resistor). A disadvantage of this solution is that I would still need to purchase a discrete laser diode module with integrated collimating lens.

3). **Laser diode with integrated collimating lens and onboard APC**. Known as an "APC laser diode" and developed by Arima Lasers Corporation in Taiwan (**www.arimalasers.com/en/products02.aspx**), these units contain a laser diode, plastic or glass collimating lens, driver, and APC circuit on an ASIC — all built into a single three-lead package (with only two of the leads actually used). Lots of different sizes, laser diode wavelengths, output powers, and collimating lens options are available, and pricing ranges from around $2 to $8 in 1K quantities depending on those options. The laser diode is simply turned on by providing 3 VDC to the module — no additional circuitry is required. The advantage here is price and integration, since the entire unit is in one package.

During my evaluation, I kept my eye on three important criteria:

- Ease of use.
- Consistency of laser spot/diameter/shape across multiple samples of the same device.
- Reliability of laser driver with a noisy input voltage (e.g., how well do the APC protection mechanisms actually work).

I ultimately selected Arima's APCD-635-02-C3-A laser diode — a Class IIIa device with a maximum power output of <= 3 mW at 635 nm. It has a metal housing with an integrated glass collimating lens. The cost of the laser diode ($7.55/1K) is essentially double the cost of the 650 nm wavelength version ($3.65/1K) that is more commonly used in low cost laser pointing devices. However, the OVM7690 — like most camera modules — contains an IR elimination/cutoff filter (more recognizable to engineers as a *low pass filter*) that blocks infrared and passes visible light.

With the filter used on the OVM7690, 50% of the

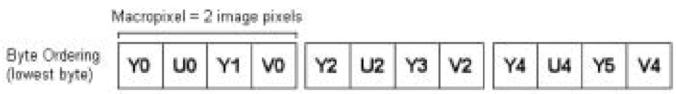


FIGURE 5. *Y0U0 Format.*

optical energy is lost at 650 nm. The shorter wavelength of the 635 nm diode means more optical power can pass through the camera's filter. More optical power means better detection of the red laser spot by the OVM7690 within an environment and, thus, a more likely successful laser range measurement. Further discussion on webcam's IR filters can be found at **http://www.david-laserscanner.com/forum/viewtopic.php?t=1337**.

For those with safety concerns, the few documented cases of eye damage with Class IIIa devices (which include — for example — most run-of-the-mill red laser pointers, laser levels, and laser-based thermometers) are related to someone staring at the beam for a prolonged period (**http://en.wikipedia.org/wiki/Laser_safety#Laser_pointers**). The laser diode for the LRF is only enabled for a single frame capture which currently takes ~400 mS.

OVM7690 Camera Interface

The Omnivision OVM7690 provides a digital interface to the host microcontroller:

- **DVP[7:0] (Digital video port):** Eight-bit wide output bus corresponding to pixel information sent in the selected output format from the OVM7690 (RAW RGB, RGB565, CCIR656, or YUV422/YCbCr422).
- **VSYNC (Vertical sync):** Indicates the beginning of a new frame by pulsing high.
- **HREF (Horizontal reference):** Indicates the start of the next row of pixels by pulsing high. By keeping count of the number of HREF pulses received since the last VSYNC, we can determine which horizontal line of the video frame we are currently on.
- **PCLK (Pixel clock):** Asserted when valid pixel data is available on the DVP bus. For a 640 pixel line in YUV422 format (16 bits/pixel), we should see 10,240 pixel clock cycles after an HREF pulse.

To help reduce intellectual property theft, OmniVision (like many other camera vendors) does not release many specific camera operating details. Documentation was thin on how the registers needed to be configured to get the camera up and running. I had to port the start-up settings from a file provided with Omnivision's PC-based evaluation tool (OVTATool) which communicates to an Omnivision camera module over a USB interface, and reverse-engineered a few additional settings by watching the bus communication between the camera and PC host.

All told, there were nearly one hundred eight-bit registers requiring configuration, including — but not limited to — general settings, output format selection, resolution, frames per second, lens correction values, color matrix values, edge and de-noise settings, Automatic Error Control (AEC), Automatic Gain Control (AGC), Automatic White Balance (AWB), gamma values, and flicker control.

I configured the camera for its maximum 640x480 VGA resolution and for data output in the YUV422 format (**http://en.wikipedia.org/wiki/YUV** and **http://en.wikipedia.org/wiki/YCbCr**). Y is the luma component — brightness in grayscale — and U and V are chroma components — color differences of blue and red, respectively. The particular format of YUV422 used by the OVM7690 is known as YUY2 (**www.fourcc.org/yuv.php**), in which each 16-bit pixel is given an eight-bit Y component and alternating eight-bit U or eight-bit V component. Y0U0 corresponds to a single pixel starting from the left, Y1V0 is the second pixel, etc. Every location has Y data, and U and V are every other pixel. Check out **Figure 5**.

Timing is Everything: My First Frame Capture

My first attempt at retrieving the digital video signal from the OVM7690 using my custom development platform looked fine in theory, but the frame grabber routine was written in Spin and too slow for the camera. I started over from scratch and re-wrote it in the more efficient Propeller Assembly (PASM).

The frame grabber cog only runs when started by a

Biography

Joe Grand is an electrical engineer and the president of Grand Idea Studio (**www.grandideastudio.com**), where he specializes in the design and licensing of consumer products and modules for electronics hobbyists. He can be reached at joe@grandideastudio.com.

calling object. It then grabs a single frame from the camera and stores it in the frame buffer. I've allocated 5,280 longs (~20.6 KB) in hub RAM for the frame buffer, leaving just under 4 KB for the stack and variables.

Grabbing a frame consists of waiting for VSYNC to go high which signals the start of a new frame, then waiting for HREF to go high which signals the start of a new line. Then, pixel data (alternating Y and U/V bytes) is captured from DVP[7:0] every time PCLK goes high and is stored in the frame buffer. After the complete frame is stored in the buffer, the cog sets a flag in hub RAM to a non-zero state so the calling object knows that the frame grab is done. The cog then stops itself.

FIGURE 6. *Giving the "thumbs up" to the camera to celebrate a successful frame grab (actual lower res image from the camera).*

I needed to increase the Propeller's system clock from 80 MHz to 96 MHz (using a 6.0 MHz crystal) in order to meet the camera's timing requirements. Even at a relatively slow PCLK of 2 MHz, I only have 0.500 µS to properly read and store each pixel. At 96 MHz — where each cycle takes 0.01042 µS — that equates to 48 cycles. Since most instructions on the Propeller take four cycles, I don't have much time to work with!

To make the timing even more tricky, the camera's image data (passed in on DVP[7:0]) is only valid when PCLK is high, so I have to read the data within 24 cycles (the first half of PCLK). Then, I have the next 24 cycles — while PCLK is low — to store the data into the frame buffer and increment counters/pointers.

Since I had yet to write any tools to aid in acquiring images from the camera, I went through a very manual process to create a bitmap image for viewing. The LRF dumped the frame in printable ASCII through its serial port to the Parallax serial terminal. Then, I copied the bytes into a hex editor, saved it as a .RAW file, imported the .RAW into Photoshop, and saved it as a .BMP. It was time-consuming, but worked just fine for development purposes. Check out a video of the process in action at **www.youtube.com/watch?v=URqUYhg4IvI**.

It took me a few days to tweak the frame grabber cog's timing (I was accidentally missing pixels, which caused a corrupted image) and to fix some incorrect automatic exposure settings (which were giving me extremely dark images). Once the bugs were squashed, I was able to grab my first correct image (see **Figure 6**). This was a major milestone of the LRF project.

Until Next Time ...

Happy with my progress thus far, the next steps are to complete the hardware design, write the image processing routines, and test the LRF with some real world measurements. I'll discuss all of that in the next article. See you then! **SV**

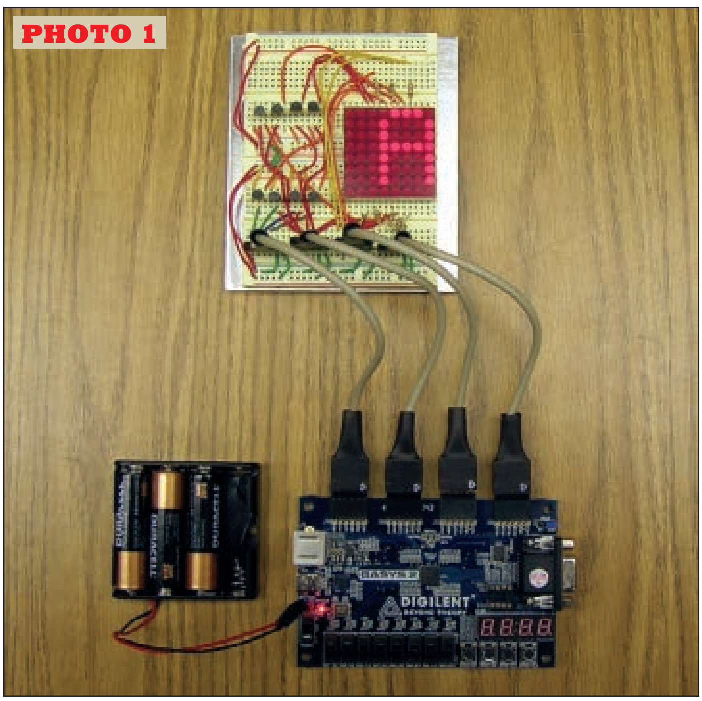
Getting Started With FPGAs — Part 2

by David Ward

In the first article about FPGAs, the reader was introduced to the Digilent Basys2 FPGA trainer and the Xilinx XC3S100E 100K gate FPGA in a 132-pin surface-mount package. The reader was shown how to enter a simple two input AND gate in VHDL, compile the listing, generate a configuration bit file, and download that bit file into the FPGA and test it.

In this second and final article, we will demonstrate a more complex and useful digital design that will use the Basys2 FPGA trainer to control a scrolling message on an 8 x 8 LED matrix display; see **Photo 1**. You will need to use four of the Digilent expansion cables and all of the 16 available FPGA expansion pins to make this circuit operate. The 8 x 8 LED matrix that is used is a Kingbright part number TC15-11SRWA 1.5" dot matrix display from **www.kingbrightusa.com** for $4.76. When you look up the Kingbright matrix, you'll notice that the top surface is not red like in **Photo 1**. If you cover the surface with red tail light repair tape, your letters appear brighter.

PHOTO 1



A complete schematic diagram is shown in **Figure 1**. The cathodes of the LEDs in the matrix are connected to the column pins which are then taken to ground through eight 2N7000 FETs. Notice that no resistors are required when using these FETs in the manner shown here. The gates of the FETs are connected to the eight column outputs from the Basys2 expansion connectors. Only one of these FETs will ever be on at one time because a ring counter will be used to sequentially scan through them one at a time over and over again at a fast rate, giving the illusion that all of the LED columns are lit at the same time. In reality, a maximum of eight LEDs will ever be on at the same time. Each FET will conduct (at the most) 80 mA at one time; eight LEDs x 10 mA each = 80 mA. According to the datasheets they are capable of conducting 200 mA.

The anodes of the LEDs are connected to the row pins on the matrix. The rows will be driven by eight FPGA pins from the Basys2 expansion connectors in series with eight 150 ohm resistors. These eight resistors are necessary to limit the current from each FPGA row pin to 10 mA. The LEDs drop 1.8V when conducting 10 mA. Therefore, 3.3V (a LVTTL logic "1") – 1.8V = 1.5V and 1.5V / 10 mA = 150 ohms of resistance. The FPGA pins can be configured in the UCF file to drive 16 mA. There will not be any problems driving the 64 LEDs in the matrix with the FPGA unless you

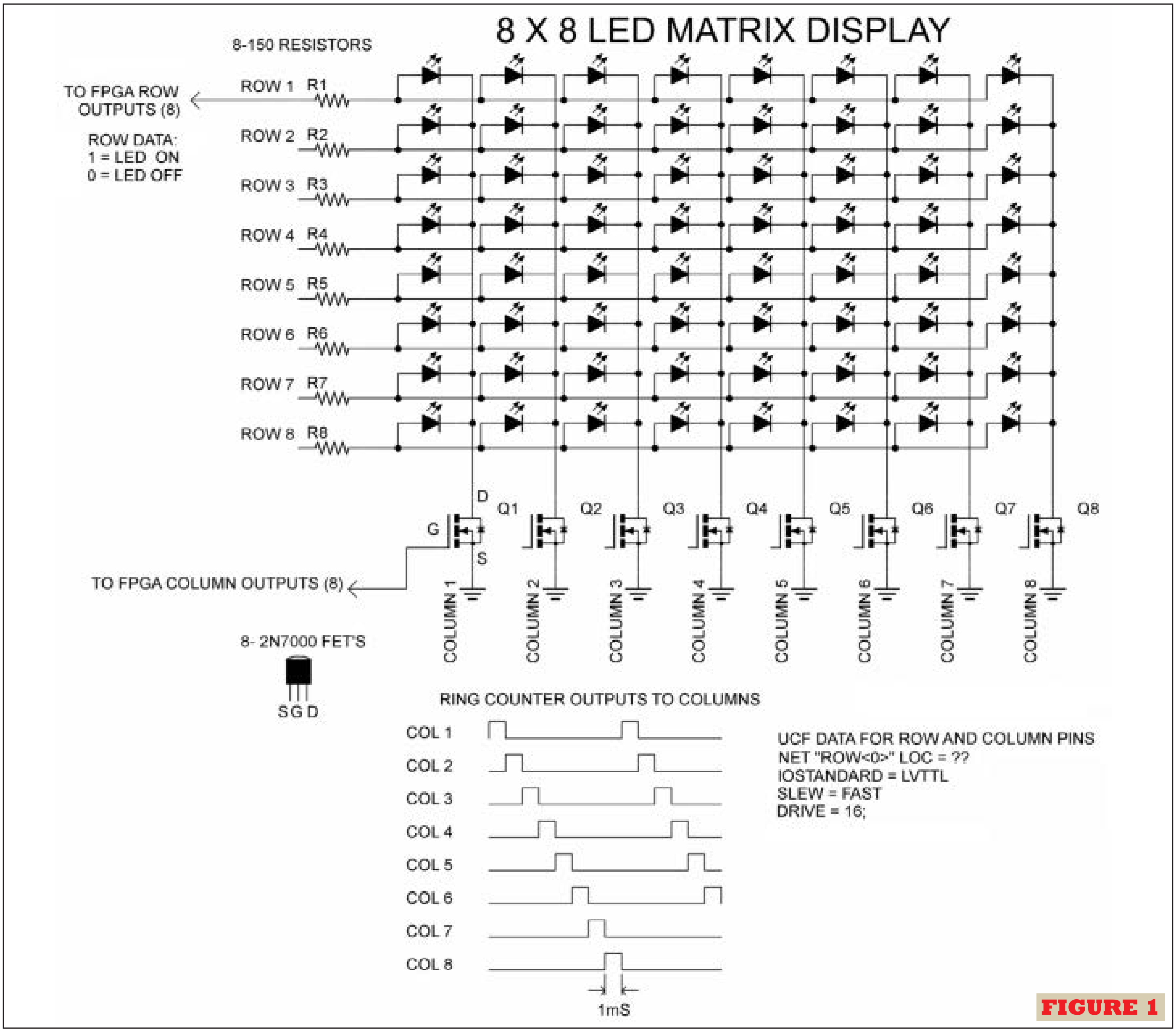


FIGURE 1

attempt to energize more than one column at a time.

Now, let's step through the VHDL and UCF listings line by line and explain what is occurring; see **Listings 1** and **2**. By the way, this code displays the message "A," "B," "C," and a blank as it scrolls from the right side of the matrix over to the left. The code has been commented here and there to help the reader see what is being done. Comments in VHDL are made by typing in two dashes ("—") and then your comments. Comments are not compiled; they are ignored by the Xilinx compiler. So, it won't be necessary to elaborate on the comment lines such as lines 1 through 3.

Lines 5 through 9 are the "Entity" section of the VHDL code listing. This is where ports or actual input and output signals are defined. Line 6 defines an input line named "clk" which is a single bit. The UCF file will direct this to pin B8 of the FPGA where the Basys2 is connected to a 50 MHz clock signal; see line 1 of the UCF file in **Listing 2**. Line 7 defines an eight-bit output port named DISPLAY_C<0> through DISPLAY_C<7>. These are the eight bits that will go out to the columns of the LED matrix. Their locations are defined in lines 3 through 10 of the UCF listing. Line 8 defines an eight-bit output port named DISPLAY_R<0> through DISPLAY_R<7>. These are the eight bits that will go out to the rows of the LED matrix. Their locations are defined on lines 12 through 19 of the UCF listing. Line 9 marks the end of the Entity section.

Lines 12 through line 56 are the "Architecture" section of the VHDL code listing. This is where the logical behavior of the circuit is defined. Lines 14 though 16 set up a signal named ABC that is 256 bits wide. It is initialized with the ones and zeros to display A, B, and C with a final blank screen. Line 17 sets up a signal called low_clk. This signal will be a lower frequency signal derived from the higher 50 MHz signal coming into the clk pin. Line 18 sets up a signal called

```
—Basys2_ABC by David A. Ward May 2011
—Displays ABC, this uses 64 bits per letter data
—and scrolls R to L
—clk info:  50MHZ = 20nS period applied to FPGA
—pin B8

entity Basys2_ABC is
        PORT (clk : IN bit;
        —50MHZ clock coming in to pin B8
        DISPLAY_C : OUT bit_vector (0 TO 7);
        —8 column outputs to ground cathodes
        DISPLAY_R : OUT bit_vector(0 TO 7));
        —8 row outputs to source LED anodes
end Basys2_ABC;

—display info is 64 bits from row1 col1 down, every
—8 bits is one column of data
architecture Behavioral of Basys2_ABC is
—256 bits, 64 bits per full display, 4 displays;
—A, B, C, and a blank
        SIGNAL ABC : bit_vector(255 DOWNTO 0) :=


"00000000011111101001000010010000100100000111111000
000000000000000000000011111110100100101001001010010
010011011000000000000000000000000000111110010000010
100000101000001001000100000000000000000000000000000
000000000000000000000000000000000000000000000000000
00";
        SIGNAL low_clk : bit;
        —reduced frequency    clock
        SIGNAL ringcounter : bit_vector(7 DOWNTO 0)
        := "00000001";
        —init ring counter

BEGIN

—clock divider portion to reduce the F of the 50MHZ
—clock down
        PROCESS (clk)
                VARIABLE inc, inc_2 : integer := 0;
                —counter variable initialized to 0
                BEGIN
                 IF(clk'EVENT AND clk = '1') THEN
                 inc
                 := inc + 1; —increment on PGT
                 IF(inc = 25000) THEN —toggle the
                 low_clk 25,000 X 20nS = 500uS
                      IF(low_clk = '1') THEN low_clk <=
                      '0'; —toggle reduced clock signal
                        inc_2 := inc_2 +1; —increment
                        scroll counter
                          IF(inc_2 = 125) THEN inc_2
                          :=
                          0; —reset counter
                              ABC <= ABC rol 8;  —time
                              to scroll, rotate data 8
                              bits left
                              ELSE ABC <= ABC;  —not
                              time to scroll, use past
                              data
                          END IF;
                              ELSE low_clk <= '1';
                              ringcounter <=
                              ringcounter rol 1;
                              —rotate ring counter
                        END IF;
                  inc := 0;      —clear count
           END IF;
END IF;

DISPLAY_C <= ringcounter;
—8 lines used to energize 8 FET's

CASE ringcounter IS
—output letter data depending on switch settings
          WHEN "00000001"  => DISPLAY_R <= ABC(255
                             DOWNTO 248);   —column 1
          WHEN "00000010"  => DISPLAY_R <= ABC(247
                             DOWNTO 240);
          WHEN "00000100"  => DISPLAY_R <= ABC(239
                             DOWNTO 232 );
          WHEN "00001000"     => DISPLAY_R <=
                             ABC(231 DOWNTO 224);
          WHEN "00010000"  => DISPLAY_R <= ABC(223
                             DOWNTO 216);
          WHEN  "00100000" => DISPLAY_R <= ABC(215
                             DOWNTO 208);
          WHEN "01000000"  => DISPLAY_R <= ABC(207
                            DOWNTO 200);
          WHEN "10000000" => DISPLAY_R <= ABC(199
                            DOWNTO 192);  —column 8
       WHEN OTHERS => DISPLAY_R <= "00000000";
     END CASE;
   END PROCESS;
end Behavioral;
```

LISTING 1

ringcounter which is eight bits wide and is initialized to "00000001." This signal will be constantly rotated left by one bit to drive the column scanning. Line 20 marks the beginning of the Architecture section after the signals have been defined.

Line 23 is the label for a process named "clk." A process operates sequentially in the FPGA instead of in a parallel manner. Line 24 defines two integer variables to be used in the process: inc and inc_2. They are both initialized to a value of zero. The first one, inc, will be incremented on every positive going transition of the 50 MHz clock. The second one, inc_2, will be incremented off of the low_clk signal to control the scrolling of the letters across the matrix. Line 25 marks the beginning of the clk process.

Lines 26 through 39 are four nested IF, THEN, ELSE statements. Line 26 will increment inc by one on every positive going transition (PGT) of the 50 MHz clock; this will occur every 20 nS. Lines 27 and 28 determine that if inc has gotten up to 25,000 or that 500 µS have elapsed, then toggle the low_clk signal. That is, if low_clk was previously a 1, then make it a 0. If it was a 0, then make it a 1. Line 29 increments inc_2 by one every time low_clk is toggled.

Line 30 determines that if inc_2 has gotten up to 125, then reset inc_2 back to zero. Then, on line 31, rotate the 256-bit signal ABC left by eight places. Line 32 determines if inc_2 has not reached 125 yet, then don't rotate ABC left eight places; leave it as it was.

Line 33 is the END IF for line 30. Line 34 is the Else for line 28 for toggling the low_clk signal. Line 35 rotates the ringcounter left by one place; this is for scanning the columns. Line 36 is the END IF for line 28. Line 37 resets the inc counter back to zero. Line 38 is the END IF for line 27. Line 39 is the END IF for line 26.

Line 41 is where the signal ringcounter is actually sent out to the eight pins of DISPLAY_C<0> through DISPLAY_C<7> which is where the matrix column scanning takes place.

Lines 43 through 53 are a CASE structure. The CASE structure will only find one of the lines from line 44 through 51 true at a time. Since the CASE structure is comparing to ringcounter to determine which line to execute, it will step sequentially through them from line 44 then on to line 45, etc., until reaching line 51. Since ringcounter is rotated every 1 mS, then the CASE structure will step through its eight choices every 8 mS. When the CASE structure finds a match — such as in line 44 when

ringcounter = "00000001" — then it will send out the upper eight bits of ABC to DISPLAY_R<0> through DISPLAY_R<7>. This is where the eight LEDs on the eight rows are energized at the anodes and the FET from column 1 is turned on to ground all of the cathodes from column 1. So, the CASE structure is where the data actually gets out to the matrix. As the CASE structure cycles through the eight ringcounter values, it will display an entire 8 x 8 image.

LISTING 2

```
NET "clk" LOC = B8;

NET "DISPLAY_C<0>" LOC = B7 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<1>" LOC = C5 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<2>" LOC = B6 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<3>" LOC = C6 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<4>" LOC = B5 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<5>" LOC = J3 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<6>" LOC = A3 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_C<7>" LOC = B2 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;

NET "DISPLAY_R<0>" LOC = A9 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<1>" LOC = A10 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<2>" LOC = D12 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<3>" LOC = B9 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<4>" LOC = C9 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<5>" LOC = C13 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<6>" LOC = C12 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
NET "DISPLAY_R<7>" LOC = A13 | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 16;
```

This happens so rapidly that the human eye cannot see it is actually scanning across the matrix and only lighting a maximum of eight LEDs at any one time. If the 256-bit signal ABC is rotated left by eight bits or one column every so often, then you can see the letters slowly scroll across the matrix from right to left. Lines 54 and 56 mark the end of the process and the Architecture sections.

If you look at the UCF listing, you'll see that it does several other things than just locate or "LOC" signals from the VHDL listing with the physical FPGA pins. The UCF also sets the IOSTANDARD to LVTTL. There are over 20 I/O standards available in this FPGA. The LVTTL stands for low voltage TTL, which means that a logic "1" is not 5.0V but 3.3V. The slew rate or the speed at which the signals transition from a 1 to a 0 and vice versa are set to FAST. They can also be set to SLOW if the receiving devices require it. Finally, the DRIVE is set to 16 mA. The drive can be set as low as 2 mA and a maximum of 16 mA.

You will probably want to make up your own messages since watching "ABC" slowly scrolling across isn't very exciting. By the way, entering all 256 of your 1s and 0s for your message isn't too convenient, but it makes the VHDL listing shorter and easier to follow. **Figure 2** shows you how the data is generated for the letters so you can develop your own messages. A blank form is included in the article downloads.

An easier way to generate your messages is to develop them column by column in Notepad and then cut and paste them into your code; see **Listing 3**. If you want a longer message, the only thing that needs to be changed is to paste your extra letters into the end of the data at line 16 and then increase the number (XXX) in line 14 by 64 for each additional character, SIGNAL ABC: BIT_VECTOR (XXX DOWNTO 0). Of course, if you don't change the numbers in the CASE section (lines 43 through 53), the message will start somewhere in the middle during the first pass through, but it will appear correct after one pass.

Hopefully, you've enjoyed these articles on FPGAs and it has given you a starting point to learn more about them and VHDL. Again, a good tutorial for VHDL is "The Low-Carb VHDL Tutorial" by Bryan Mealy available on the Internet. **SV**

```
A
00000000
01111110
10010000
10010000
10010000
01111110
00000000
00000000

B
00000000
11111110
10010010
10010010
10010010
01111100
00000000
00000000

C
00000000
01111100
10000010
10000010
10000010
01000100
00000000
00000000
```

LISTING 3

FIGURE 2

8 X8 LED MATRIX FOR LETTER "A"

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8		D7	D6	D5	D4	D3	D2	D1	D0
Row 1				*					Row 1 data	0	0	0	0	0	0	0	0
Row 2			*		*				Row 2 data	0	0	1	1	1	1	1	0
Row 3		*				*			Row 3 data	0	1	0	1	0	0	0	0
Row 4		*	*	*	*	*			Row 4 data	1	0	0	1	0	0	0	0
Row 5		*				*			Row 5 data	0	1	0	1	0	0	0	0
Row 6		*				*			Row 6 data	0	0	1	1	1	1	1	0
Row 7		*				*			Row 7 data	0	0	0	0	0	0	0	0
Row 8									Row 8 data	0	0	0	0	0	0	0	0

"B"

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8		D7	D6	D5	D4	D3	D2	D1	D0
Row 1		*	*	*	*				Row 1 data	0	0	0	0	0	0	0	0
Row 2		*				*			Row 2 data	1	1	1	1	1	1	1	0
Row 3		*				*			Row 3 data	1	0	0	1	0	0	1	0
Row 4		*	*	*	*	*			Row 4 data	1	0	0	1	0	0	1	0
Row 5		*				*			Row 5 data	1	0	0	1	0	0	1	0
Row 6		*				*			Row 6 data	0	1	1	1	1	1	0	0
Row 7		*	*	*	*				Row 7 data	0	0	0	0	0	0	0	0
Row 8									Row 8 data	0	0	0	0	0	0	0	0

"C"

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8		D7	D6	D5	D4	D3	D2	D1	D0
Row 1			*	*	*				Row 1 data	0	0	0	0	0	0	0	0
Row 2		*				*			Row 2 data	0	1	1	1	1	1	0	0
Row 3		*							Row 3 data	1	0	0	0	0	0	1	0
Row 4		*							Row 4 data	1	0	0	0	0	0	1	0
Row 5		*							Row 5 data	1	0	0	0	0	0	1	0
Row 6		*				*			Row 6 data	0	1	0	0	0	1	0	0
Row 7			*	*	*				Row 7 data	0	0	0	0	0	0	0	0
Row 8									Row 8 data	0	0	0	0	0	0	0	0

The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

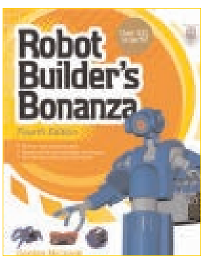
CD-ROM SPECIALS

ROBOTICS

Robot Builder's Bonanza, Fourth Edition

NEW!

by Gordon McComb

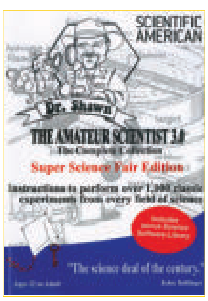


Robot Builder's Bonanza, Fourth Edition includes step-by-step plans for a number of motorized platforms. The book is described as a compendium of robotics topics, containing more than 100 projects, including 10 robot designs new to the fourth edition. These modular robots are low cost, and are made to be reproduced by readers with no training in mechanical construction.

$29.95*

The Amateur Scientist 4.0 The Complete Collection

by Bright Science, LLC



There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg $26.95 Sale Price $23.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts

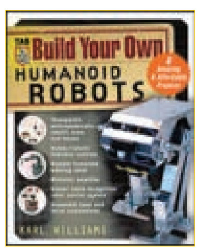
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

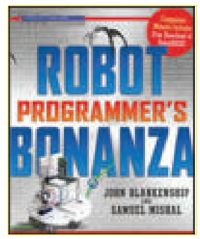


This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.

$24.95*

Robot Programmer's Bonanza

by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!

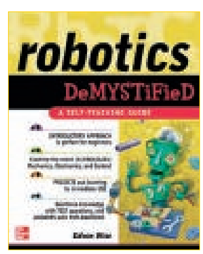
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!

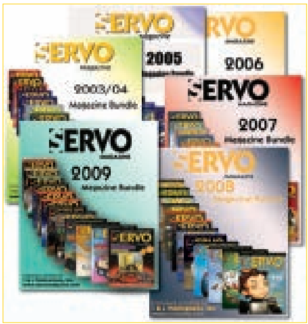


Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

**We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.**

To order call 1-800-783-4624

SERVO Magazine Bundles



Published by T & L Publications, Inc.

$57 per bundle

Save $10 off the normal price!!

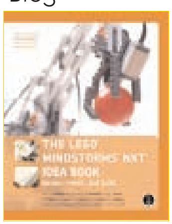
Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Call my Webstore and you'll get someone in AMERICA!

Visit my online store @ www.servomagazine.com

LEGO MINDSTORMS NXT Idea Book

by the Contributors to The NXT Step Blog

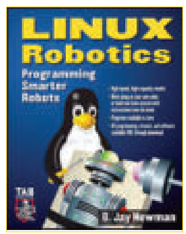


If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog — the authoritative online source for MINDSTORMS® NXT information and advice — has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

Reg $29.95 Sale Price $24.95

Linux Robotics

by D. Jay Newman



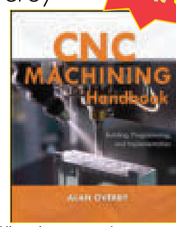
If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

$34.95

CNC Machining Handbook: Building, Programming, and Implementation

by Alan Overby

NEW!

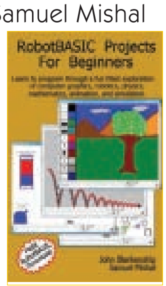


The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

SPECIAL OFFERS

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal



If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price $14.95 Sale Price $9.95

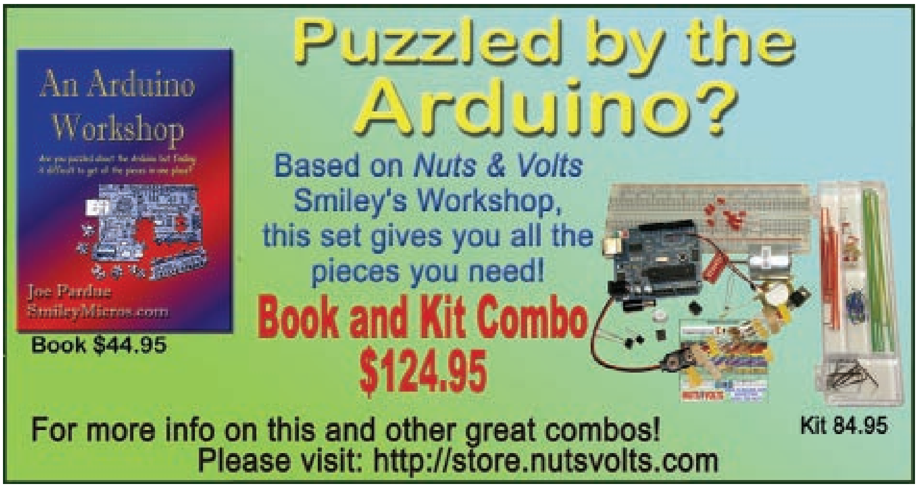
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

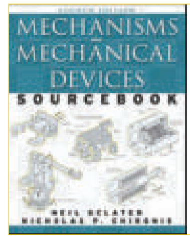
Regular Price $79.95 **Subscriber's Price $75.95**

Or order online www.servomagazine.com

SPECIAL OFFERS



Mechanisms and Mechanical Devices Sourcebook

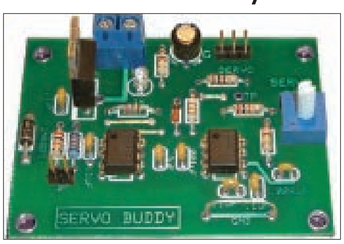


by Neil Sclater, Nicholas Chironis

Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg $89.95 Sale Price $69.95**

PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

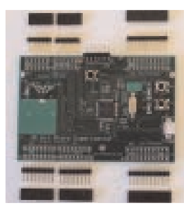
Includes an article reprint.

Subscriber's Price **$39.55**

Non-Subscriber's Price **$43.95**

32-Bit Micro Experimenter Board

The new 32-Bit Micro Experimenter is the fastest way to learn 32-bit microcontrollers.



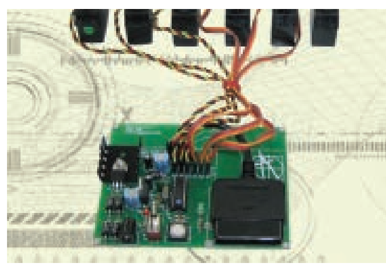
NEW!

The kit includes onboard 46 programmable I/O and USB, free software, carefully documented step-by-step experiments for USB, embedded web server, graphics and audio, wireless, RTOS, and file I/O. User pushbuttons, LEDs, and 32 kHz clock crystal. Can be used in solderless breadboard environment or stand-alone. Also supports Arduino compatible interface.

Subscriber's Price **$89.95**

Non-Subscriber's Price **$93.95**

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

Subscriber's Price **$79.95**

Non-Subscriber's Price **$84.95**

Adding A Microcontroller to The Beginner Bot

Part 3

by Gordon McComb

The study of robotics is a lifelong journey. Each new robot is a new adventure, and a fresh chance to expand your learning. For the last several issues, I've written about the Beginner Bot: an affordable and expandable platform that explains basic robotic concepts in easy steps.

In the first installment, you learned how to construct the Beginner Bot platform using wood or plastic, and how to steer it using mechanical switches from a tethered control panel. From there, you learned how to replace the switches with fully electronic influence, adding twin "eyes" and a simple one-chip brain so that your bot could follow the beam of its master's flashlight.

In this article, you'll discover how to add a low cost microcontroller, replacing hard-wired control circuitry with software programming. By moving up to a miniature computer to operate your Beginner Bot, you'll be able to modify its action and behavior just by rewriting a few lines of code.

Using the PICAXE AXE401 Development Board

I'm a sucker for ease of use, but I also like to save a dime here and there. The AXE401 development board for the PICAXE nicely combines both. The '401 is form factor-compatible with the Arduino Uno, and even accepts expansion shields designed for the Arduino.

Like the Arduino, the board provides 20 input/output (I/O) pins, six of which can serve as both analog inputs or standard digital I/O. Instead of using an Atmel ATmega328P microcontroller — as the Arduino does — the AXE401 uses a PICAXE 28X2.

If you're not yet familiar with it, the PICAXE family of microcontrollers is based on various versions of the Microchip PIC, pre-coded to permit easy programming using a Basic-like language. All the PICAXE chips come with numerous built-in features handy for robotics. These features include remote control decoding, I^2C interfacing, R/C servo operation, sound and tone making (through an external speaker), and — for the more advanced chips like the 28X2 — multi-program storage. With the latter, you can store up to four separate programs in the chip's memory and run them at will.

As with all microcontrollers, you program the PICAXE from your computer. You then download these programs to the PICAXE via a connecting cable. Free software is provided for building and downloading PICAXE programs. Visit the PICAXE website at **www.rev-ed.co.uk/picaxe** for details. There are versions of the programming editor (and other tools) for Windows, Macintosh OS X, and Linux.

The PICAXE controllers are affordably priced, starting at about $3. The AXE401 comes in kit form and combines the PICAXE 28X2, circuit board, connectors, and all the electronics. The cost is about $18. The '401 only uses through-hole parts. Soldering isn't difficult, and takes about 20 minutes. A completed AXE401 board is shown in **Figure 1**.

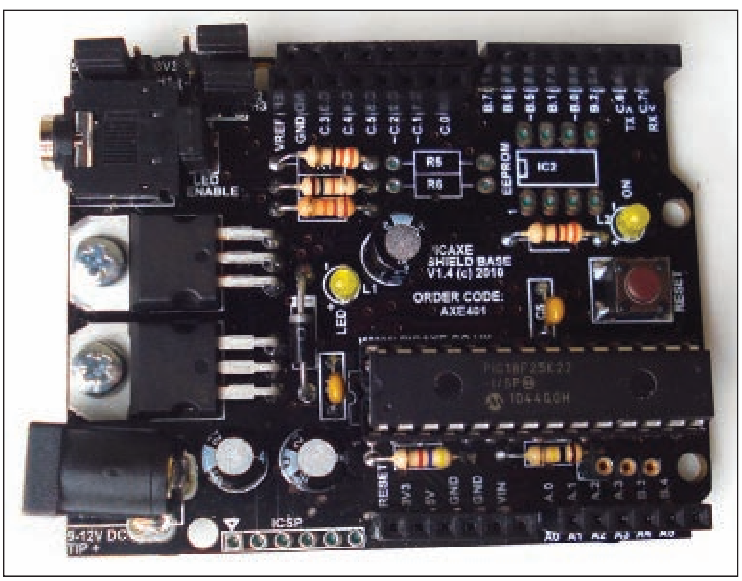


FIGURE 1. The AXE401 development board contains a PICAXE 28X2 microcontroller, voltage regulators, and assorted electronics for quick and easy hookup to your projects.

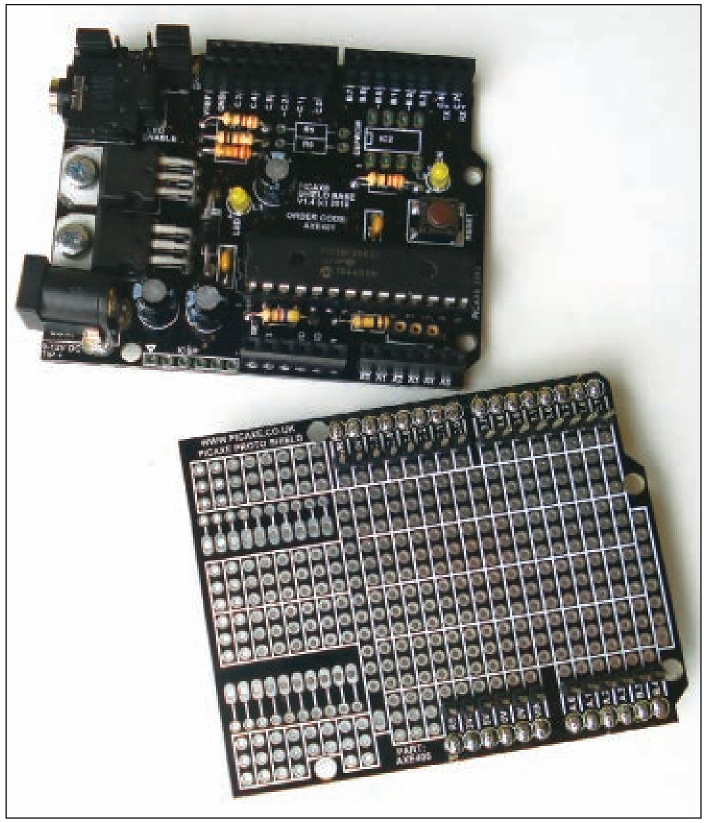


FIGURE 2. The prototyping board lets you stack or sandwich a shield on top of the AXE401. In the middle of the shield, you can place a mini solderless breadboard for rapid prototyping.

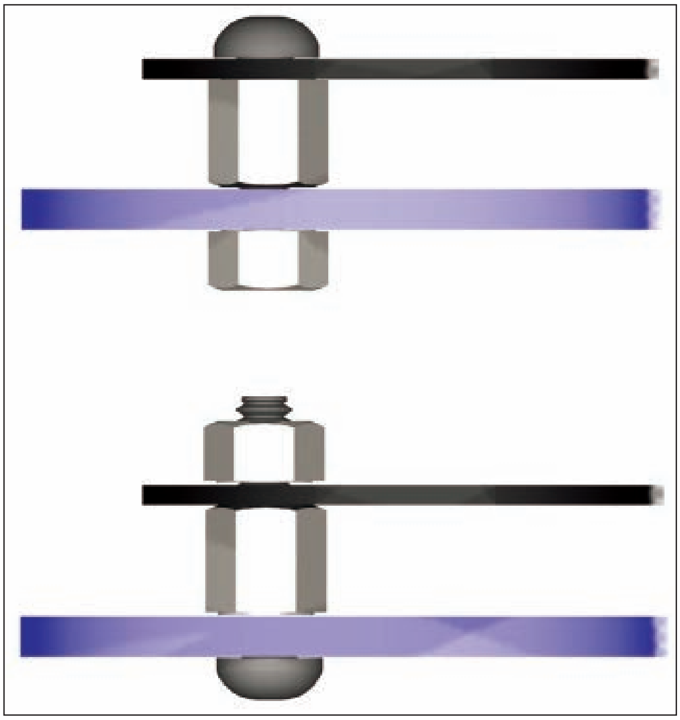


FIGURE 3. Attach the AXE401 board using jackscrews. You can insert the male threaded end of the jackscrew from the top or bottom of the '401 board. You can also use plastic or metal standoffs.

Tip! You can read more about the PICAXE family of processors — including the AXE401 — in Ron Hackett's columns in *Nuts & Volts*, the sister publication to *SERVO*. Each month, Ron explores the world of the PICAXE, and demonstrates various programming concepts and projects you'll want to try. Be sure to also read the three-part PICAXE manual that's provided in PDF format. It's included in the programming editor software download.

You need a download cable to program the 28X2 chip on the '401 board. Two versions of cable are available from the PICAXE folks, or you can make your own if your computer has an older RS-232 serial port. Ready-made download cables are available for RS-232 serial (about $6) and USB.

The cost of the USB cable is about $25, but keep in mind you only need one cable, no matter how many PICAXE chips or development boards you have. This makes the PICAXE particularly attractive in classroom settings, where the one download cable can be shared among all the students. The cable is only needed when downloading programs to the PICAXE. Once the program is downloaded, the cable is removed from the AXE401 board, so that the Beginner Bot can trawl around on its own.

Remember that to use the USB cable, you need to first install the PICAXE USB drivers. They're included with the programming editor software. The USB drivers have their own installation program which you should run before plugging in the cable. Once the drivers are installed, the USB cable will appear as a serial communications port to your computer.

Included with the AXE401 is a bare prototyping shield. It's intended for expanding the '401 with additional external components. It sandwiches with the main board via a set of header pins as shown in **Figure 2**. I'm showing the board with two sets of male headers: one set points down and mates with the AXE401 board; and the other set points up, allowing connection to a mini solderless breadboard on top. I'll discuss connecting the AXE401 to a solderless breadboard later in this article.

Mounting the AXE401 Board on the Beginner Bot

Use the second "deck" we added in Part 2 of this series for mounting the AXE401 development board. If you followed along and built that version, you'll want to carefully remove the wiring and components from the mini solderless breadboard and put them aside — you'll be reusing some of the parts.

Remove the second deck from Beginner Bot and drill two holes to mount the AXE401. Pick the holes in the upper right and bottom left of the board. Center the board on the deck (but leave a little more space toward the front) and mark the position for the two holes with a scribe or small nail. Carefully drill new holes using a 1/8" drill bit.

Use a pair of 4-40 jackscrews, nuts, and 4-40 x 1/4" machine screws to mount the AXE401. Jackscrews are like miniature standoffs with male threads on one end, and female on the other. If you used 1/4" thick material for the

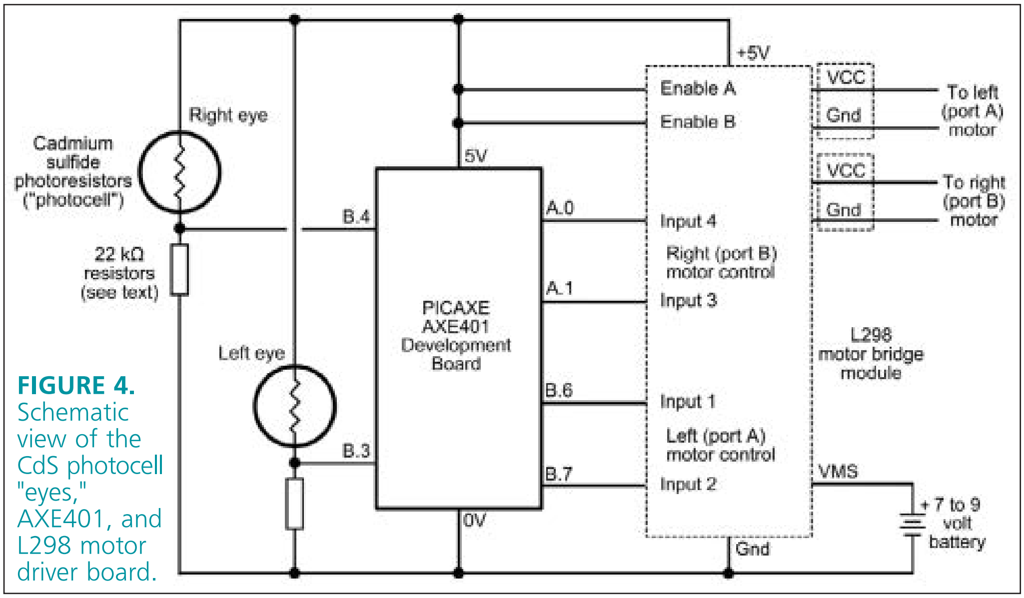


FIGURE 4. Schematic view of the CdS photocell "eyes," AXE401, and L298 motor driver board.

come with self-adhesive foam tape. For my prototype, I just soldered in some additional header pins front and back to keep the board in position. I prefer to not use foam tape, so that I can easily replace the breadboard or use it for something else.

second deck, be sure the threads of the jackscrew are long enough to catch a nut on the other side. Or, you can simply turn the jackscrew upside down and mount it so that the male threads point up. See **Figure 3** for both arrangements.

If you use metal jackscrews and fasteners, be sure to add plastic washers to prevent a possible short circuit. The washers are probably not needed, but they're handy just in case a fastener gets too close to an exposed circuit part.

No jackscrews handy? Most any 1/4" to 1/2" metal or plastic threaded standoff will work. These are sold by numerous online retailers, including All Electronics, Pololu, and others. Pick those made for #4 hardware; the holes in the AXE401 board won't accommodate the larger stuff.

Once the AXE401 board has been mounted, you can attach the prototyping shield on top with the mini solderless breadboard in the middle. Most mini breadboards

Wiring for Motor Control

Phase 1 of the Beginner Bot project used switches for manual control of the bot's motors. Then in the second installment, you discovered how to replace the switches with an L298-based H-bridge module. This module is controlled by a simple one-chip electronic circuit to allow the robot to follow a light beam striking two photocells.

By using the PICAXE microcontroller, you can command your Beginner Bot to follow your programmatic instruction. Instead of rewiring sensors and inputs, you can quickly and easily modify the behavior of the Beginner Bot simply by altering a few lines of code.

See **Figure 4** for how to connect the AXE401 development board to the L298 H-bridge module. Use the mini breadboard as an interface between the motor bridge, the two photocells, and the AXE401 board. **Figure 5** shows the circuit in breadboard view.

The jumper wires you use between the AXE401 and the breadboard will depend on the kind of header pins you've installed on the protoboard shield. For my prototype, I used standard male header pins which require male-female jumpers. I'm using SchmartBoard #920-0022-01 5" male-female jumpers. You can also create your own using crimp-on breadboard pins (Electronix Express #03BBPINS), with your choice of 22 or 24 gauge stranded conductor any length you wish.

Another method is to use two sets each of 1x6 and 1x8 female headers, then connect to the breadboard with the more commonly available (or made) male-male jumpers. One such product is Pololu #1016. You'll still need the male headers underneath to connect the shield to the AXE401 board.

For an all-in-one solution, there's the "stackable" header which combines long male header pins on the bottom with female connectors on the top. These are available among numerous sources, including

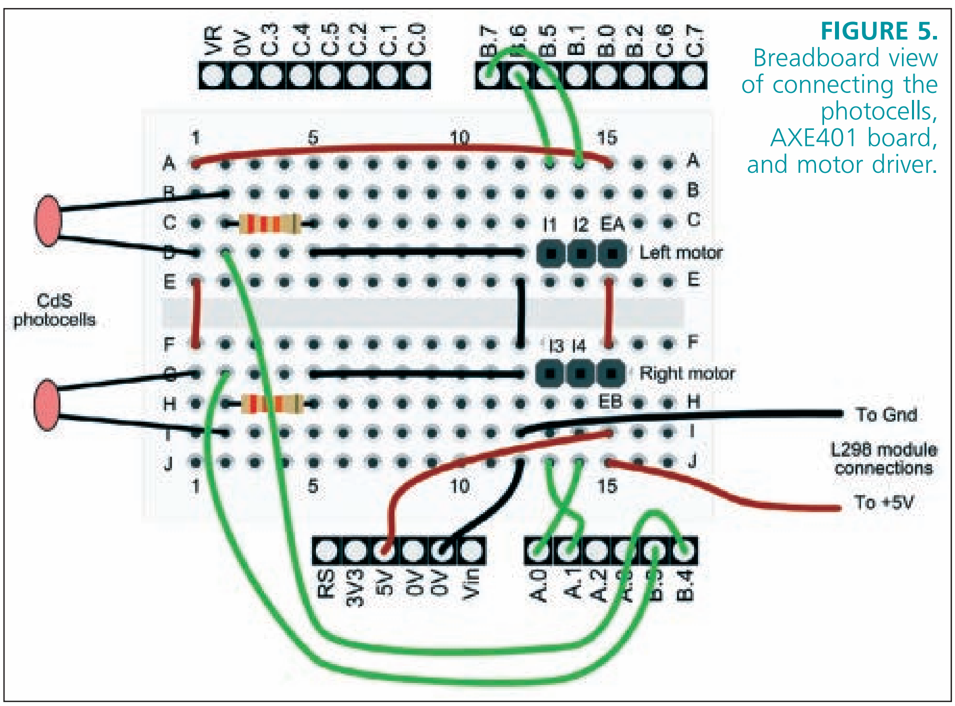


FIGURE 5. Breadboard view of connecting the photocells, AXE401 board, and motor driver.

SparkFun (#PRT-10007) and Adafruit (item #85). See the **Sources** box website addresses for these and other online retailers.

Here's how the circuit works: Two cadmium sulfide (CdS) photocells detect the amount of light falling on them. This type of photocell exhibits a change of resistance depending on the amount of light: The less light, the higher the resistance; the more light, the lower the resistance. For each CdS "eye," a 22 kΩ resistor turns the resistive output to a varying voltage — the CdS cell resistor and the fixed resistor form a voltage divider circuit.

The voltage produced at the junction between these components stretches from between zero and five volts. The outputs of the sensors are connected to two of the AXE401's analog inputs — the pins marked B.3 and B.4.

The value of 22 kΩ for the resistors connected to each CdS cell is determined experimentally. There are no standards in CdS photocells, and their dark and light resistance can differ greatly — even among components of the same type. You'll want to try different values to determine the best sensitivity for the photocells you use. You want the highest sensitivity while maintaining the widest possible swing between zero and five volts.

(Note: The pin IDs reflect the nomenclature used on the 28X2 chip itself, and are printed on the AXE401 board. For your reference, the *B* is a *port* on the chip containing many pins, and the *3* or *4* is a specific pin on that port. The 28X2 has three ports, labeled A, B, and C. There are different numbers of pins available on each of the ports.)

Recall from last month that the L298 H-bridge module requires two inputs per motor. The direction of the motor is determined by the instantaneous value of these two inputs, according to **Table 1**.

By setting the pins LOW (zero volts) or HIGH (five volts) in programming, you can control the operation and direction of either motor. You'll see exactly how this is done in the next section.

You may have noticed that the AXE401 contains its own power plug and five volt voltage regulator — it even has a second regulator for 3.3 volts. For this project, we won't be using these features, as the Beginner Bot instead draws its five volt power from the regulator on the L298 H-bridge that we've selected for the project. This power drives the *logic* portion of the L298 module, and also operates the AXE401.

This power connection arrangement simplifies the wiring, but know that when it comes time to reuse your AXE401 for some other project that it's capable of being separately powered using its own onboard voltage regulator.

TABLE 1

Input A	Input B	What Happens
Low	Low	Motor stops
Low	High	Motor turns one direction
High	Low	Motor turns the other direction
High	High	Motor stops

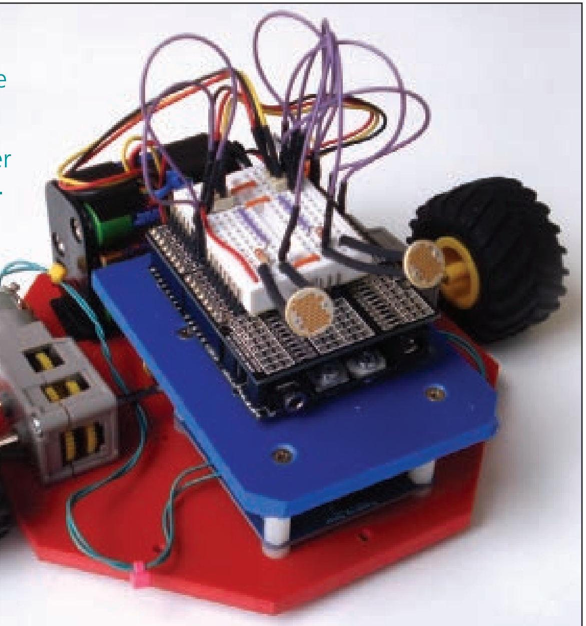


FIGURE 6. Completed Phase 3 of the Beginner Bot, with PICAXE microcontroller added on top.

Figure 6 shows the completed *Phase 3* of the Beginner Bot, with AXE401 board, prototype expansion shield, and populated mini solderless breadboard.

As with the *Phase 2* version of the Beginner Bot that demonstrated control using a hex inverter IC, the two CdS photocells are mechanically attached to the front of the mini solderless breadboard and poke out like snail's eyes.

Sources

Precut and predrilled Beginner Bot base, with all construction hardware:

Budget Robotics
www.budgetrobotics.com

PICAXE documentation, software, sales (UK and EU):

PICAXE Home
www.rev-ed.co.uk/picaxe

Tech Supplies
www.techsupplies.co.uk

AXE401 development board, serial and USB download cables for PICAXE:

HVW Tech*
www.hvwtech.com

PH Anderson
www.phanderson.com

Robotshop*
www.robotshop.com

SparkFun*
www.sparkfun.com

Mini solderless breadboards, jumper wires, header pins, etc.:

Adafruit Industries
www.adafruit.com

All Electronics
www.allelectronics.com

Electronix Express
www.elexp.com

HVW Tech*
www.hvwtech.com

Jameco Electronics
www.jameco.com

Mouser Electronics
www.mouser.com

Pololu
www.pololu.com

RobotShop
www.robotshop.com

Schmartboard
www.schmartboard.com

SparkFun
www.sparkfun.com

Solarbotics
www.solarbotics.com

*At the time of this writing, these resources were not yet carrying the AXE401, but they do stock other PICAXE parts. Check for current availability.

Use the Right Motors!

The Beginner Bot uses a pair of Tamiya gearboxes that have been modified according to instructions provided in Part 2 of this series. Specifically, the motors used in the gearboxes have been replaced with versions that provide for operation at six to 12 volts, and with higher efficiency. These motors are available from Pololu (item #1117), among other sources. Cost is under $2 each.

Be sure to not use the stock motors that come with the Tamiya gearboxes. These are rated for only three volts and can consume copious amounts of current. This current exceeds the rating of the L298 H-bridge used to control the motors.

LISTING 1

```
#picaxe 28x2            ; Specify 28X2 PICAXE chip

main:                   ; Main program loop


  ; Go through series of motions
  gosub robot_forward
  pause 2000

  gosub robot_backward
  pause 2000

  gosub robot_right
  pause 2000

  gosub robot_left
  pause 2000

  gosub robot_stop
  pause 2000

  goto main             ; Repeat main program

robot_forward:          ; Motion control routines
  high B.6              ; Right motor controlled by
  low B.7               ;   pins B.6 and B.7
  high A.0              ; Left motor controlled by
  low A.1               ;   pins A.0 and A.1
  return

robot_backward:
  low B.6
  high B.7
  low A.0
  high A.1
  return

robot_right:
       high B.6
       low B.7
       low A.0
       high A.1
       return

robot_left:
       low B.6
       high B.7
       high A.0
       low A.1
       return

robot_stop:
       low B.6
       low B.7
       low A.0
       low A.1
       return
```

Gently bend the leads of the cells so that they point slightly upward and outward, like that in **Figure 7**. I've added some unshrunk heat shrink tubing over the photocell leads to provide both insulation from short circuits and an extra bit of mechanical support.

Testing Motor Operation

Listing 1 shows a demonstration program for testing the basic operation of the AXE401 board, the H-bridge, and the motors. Type or download this program from the *SERVO* website, then:

1. Place a small block under the Beginner Bot base to lift the wheels off your worktable.
2. Connect the battery to apply power to the H-bridge and AXE401 board.
3. Connect the programming cable between your PC and the AXE401, and be sure its communication port is selected in the PICAXE programming editor (choose *View>Options>Serial Port*).
4. Download the program to the AXE401. You'll be prompted if there are connection errors.

The downloaded program starts automatically. Assuming the motors have been connected properly, both motors should turn the same direction forward and back. The motors will stop after one cycle. You can press the Reset button or momentarily break power from the batteries to rerun the motor demonstration.

If one or both motors turn in the wrong direction, remove power and flip the terminal wiring from the affected motor on the H-bridge.

Let There Be (Flash) Light

In the previous installment, you learned how to control the Beginner Bot using a flashlight, shining the light into the photocell eyes. The simple circuit depicted in this article reacted to the bright light, steering the robot toward it.

Listing 2 extends the concept; this time, in a purely programmatic way. The program tells the PICAXE microcontroller to read the value from both photocells. A series of *If* tests determine if there's enough light to follow, and if so, in what direction the robot should travel. This is a good example of conditional logic.

The program first sets a threshold value to determine the boundary between dark and light. I've set this value to 180 — out of a range of 0-255 — as a starting point. Try higher or lower values to see what works best with your particular CdS cells.

When both cells receive light over the threshold, the robot drives forward. When only one cell receives light over the threshold, the robot turns in the direction of the light. If neither cell receives light over the threshold, the robot stops.

Let's test all this. Download the program in **Listing 2**, and when the download is complete remove the

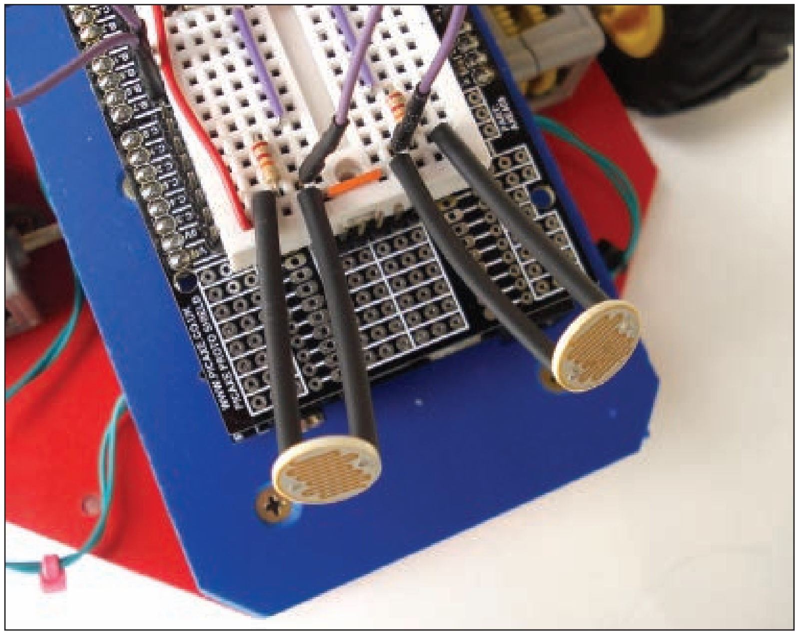


FIGURE 7. Spread out the sensor surface of the photocells so that you can direct the beam of a flashlight into either both or just one at a time.

programming cable. Move to a darkened room, apply power to the robot, and place it on the ground — tile or wood floor is better than thick carpeting due to the limited clearance under the Beginner Bot.

To start, aim a bright flashlight (preferably one with a strong narrow beam) *away* from the Beginner Bot. Both motors should remain off. Now, shine the flashlight directly into the photocells. The robot should move toward you.

Get close to the robot and aim the flashlight into just one photocell; you may need to gently spread the cells apart if they're too close together. The robot should turn toward the photocell with the light shining into it.

If your robot moves when there's no light falling on the CdS cells, try changing the threshold value to something higher. Conversely, if the light from the flashlight seems to make no difference, enter a lower threshold and try a darker room. Keep in mind that any significantly bright light source will "blind" the robot to the flashlight.

If you're still not getting results, try the simple test program in **Listing 3**. When run, it displays the numeric values obtained from both CdS photocells in the PICAXE editor's Debug window. Keep the download cable attached to the AXE401 during this test, and note the values in the *b0* and *b1* boxes (upper left corner). Aim the CdS cells at a bright light source and the value should climb high — up to the maximum of 255. Block all light and the value should fall close to zero.

If nothing like this happens, double-check your wiring. Try different CdS cells, and re-run the test.

LISTING 2

```
#picaxe 28x2
symbol threshold = 180       ; Define threshold between
                             ;  dark and light

main:                        ; Main program loop

  readadc B.3, b0            ; Read analog pins B.3 & B.4
  readadc B.4, b1

  ; Determine direction of robot based on sensor input
  if b0 > threshold and b1 > threshold then gosub robot_forward
  if b0 > threshold and b1 < threshold then gosub robot_left
  if b0 < threshold and b1 > threshold then gosub robot_right
  if b0 < threshold and b1 < threshold then gosub robot_stop

goto main                     ; Repeat program loop

(repeat motor control routines from Listing 1 here)
```

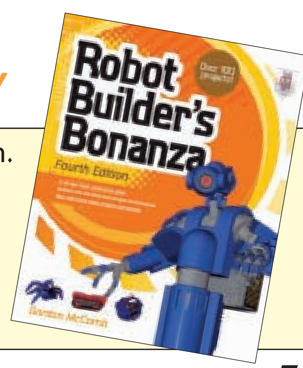
LISTING 3

```
#picaxe 28x2
main:
  readadc B.3, b0
  readadc B.4, b1
  debug b0
  goto main
```

Next Up: Arduino-based Beginner Bot

The Beginner Bot isn't tied to just one kind of microcontroller. It's a universal design that lets you use your choice of microcontroller. I wanted to start with the PICAXE — and especially the AXE401 development board — because it's inexpensive and easy to use. You're free to try other microcontrollers, like the Arduino, BASIC Stamp, or Parallax Propeller.

In the next installment, we'll do just that. You'll see how to connect the Beginner Bot to the popular Arduino Uno development board, and how to adapt the light seeking abilities of your little robot to Arduino programming code. You'll also discover how to add touch sensors and other features. **SV**



Gordon McComb is the author of *Robot Builder's Bonanza*, now in its fourth edition. Greatly expanded and updated, this best selling book covers the latest trends in amateur robotics, and comes with 10 all new robot construction projects, plus more ideas for building robots from found parts. Look for *Robot Builder's Bonanza, 4th Ed* in the *SERVO* Webstore at **http://store.servomagazine.com**. Gordon may be reached at rbb@robotoid.com.

Then and NOW

HOW ROBOTICS HAS CHANGED OVER THE YEARS

by Tom Carroll

It is always interesting to read articles about robotics in non-robotics magazines or hear people talk of the changes in the field. Quite often, when people realize that I write about the history of robotics, I invariably hear the comments "I remember when robots ..." or "Robotics has really changed since ..." I sometimes answer with "The times, they are a changing," and give them a short synopsis of my thoughts on the subject. The print and TV media has certainly expressed their ideas on the subject of robotics and the many changes over the years.

The August '11 issue of *National Geographic* featured an article entitled "Us and Them — Robots Get Real." The article went on to state that "sophisticated robots may soon cook for us, fold our laundry, even babysit our children." In the June issue of *Control Engineering*, an article entitled "The Changing Face of Robotics" mentioned the extremely popular FIRST robotics competitions as a way to entice young people into the fields of science and technology. The article went a bit further to speak on the emergence of mechatronics, new delta-style pick and place industrial robots, and autonomous UAVs and UGVs.

The non-technical media has taken notice on just how rapidly robots have entered our lives and are changing the way that we live. This column centers on how robotics has changed over the years, so these types of articles always catch my eye. The IEEE *Spectrum* magazine recently headlined: 'Next Big Thing in Silicon Valley: Robotics?' Are we at some amazing turning point in this science? Are the strictly computer and semiconductor manufacturers moving into robotics and automation, or are these key industries spawning these new companies? Is the heartbeat of robotics innovation moving from the east coast areas of MIT in New England and Carnegie Mellon in Pittsburgh to the birthplace of microprocessor-based computers — the San Jose area in California? I believe it is a bit of all three.

Other rapidly growing areas of technology such as Austin, TX and the Research Triangle Park in North Carolina are riding the latest boom, though an article in July's *Fortune* by David Kaplan is warning of a potential 'Tech Bubble 2.0.'

Robot Industries Shift Away From Michigan And The Auto Manufacturers

Let's step back several decades and view the state of robotics at the start of this new industry. The big push in robotics in the '60s, '70s and through the '80s was industrial robots and their implementation in the booming automobile manufacturing industries centered around Detroit. My employer (Rockwell) asked me to study how robotics could be used in the aerospace industry. I went to the yearly robot conferences that were sponsored by Robotics International of the Society of Manufacturing Engineers. RI/SME is based in Dearborn, MI and produced robot conferences and exhibits that alternated each year between Detroit and Chicago. Hundreds of robot manufacturers — both large and small — were scattered around the suburbs of Detroit and nearby Canadian cities. The exhibit halls in Detroit and Chicago were crammed full of robots: industrial, educational, experimental, and whatevers. Thousands of professionals, members of the media, university researchers, and some who were just interested in looking at what was new roamed the aisles, looking at the future. Local TV news crews conducted interviews with manufacturers of especially cool looking robots, and maybe a noteworthy spokesperson such as Joe Engelberger. If it had anything to do with robots, it was at one of these RI/SME conferences either as an exhibit or the title of one of the technical sessions.

Most of the speakers were from the manufacturing sector, either representing a robot manufacturer and touting the many unique applications that their products could perform, or the user segment from industry that spoke of the positive ROI (return on investment) and time savings of using robots. News people rarely interviewed those of us in the other robot interest groups. In the early '80s, the exhibit aisles were filled with various sizes and styles of robots, most from US companies. Soon, the large Japanese companies began to show their superior robots and, by the '90s, most of the American companies manufacturing industrial robots were either bought out by Japanese, Swedish, or German robot manufacturers, or just quietly faded away into oblivion.

My interests were primarily robotic applications for the space industry and for actual in-space deployment such as space station applications and Mars/moon rovers. There was a core group of us from other companies and industry groups that gave talks on non-manufacturing robot applications. Some of the manufacturers of the early home robots (such as RB Robot, Heathkit, and Androbot) were shunned a bit as they did not fit the mold of a 'real' robot. Our special interest group presentations were well attended, but nothing like the large groups attending the industrial robot sessions.

Some of my presentations on space station robotics managed to keep most of the group of 50 or so people from falling asleep, but it was the presentations from Odetics and their six-legged, human-sized robot that really kept a 100 or more people on the edge of their seats. The Odetics Odex 1 from around 1983 is shown in **Figure 1** and was one of the most interesting robots that I've ever seen. Standing a bit over five feet tall, this robot was shown climbing out of the bed of a small truck, and then turning and lifting the back of the truck off the ground. It was designed as a possible hazardous duty robot, working in nuclear power plants and the like, but the marketing aspect never generated any sales. It was ahead of its time.

FIGURE 1. The Odetics Odex 1.

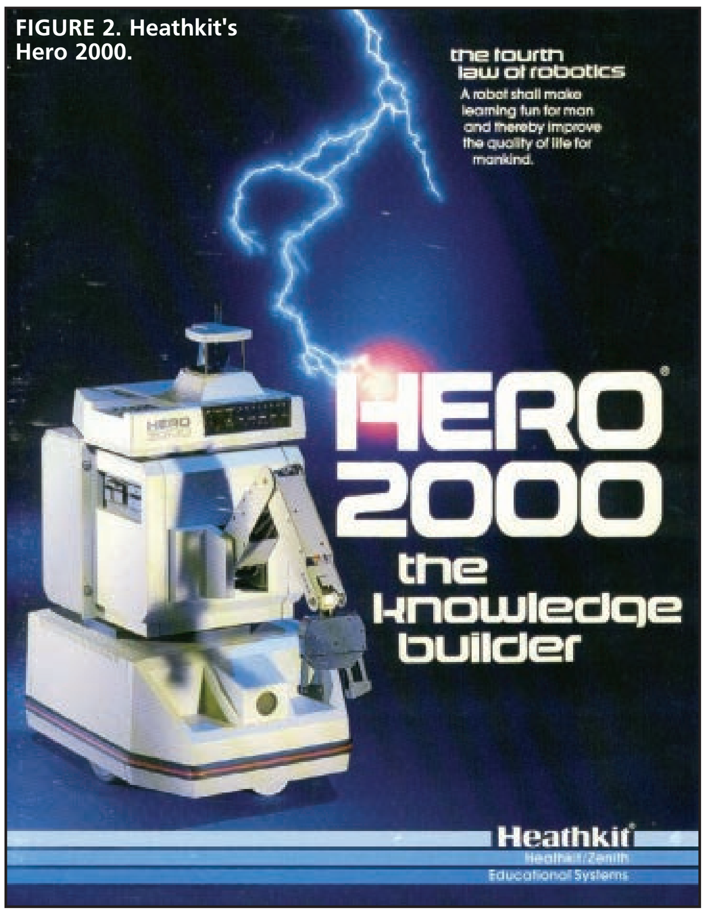


FIGURE 2. Heathkit's Hero 2000.

National Service Robot Association Has Difficulties In The Late '80s

After a period of several years highlighting only the industrial robot sector, the RI/SME began to realize that the other segment of robotics was garnering a lot of interest and attention. This 'other' segment included service robots, military robots, police robots, UAVs, and types of robots that defied definition. Several of us got together to form the National Service Robot Association (NSRA) and Doug Bonham of Heath was elected chairman. Heath was the manufacturer of the very popular Heathkit Hero (Heath Educational RObot) series of robots back in the '80s and Bonham was the ideal board chairman. To this day, any of the Hero educational robots are sought-after collector's

FIGURE 3. The hot-selling Microsoft Kinect.

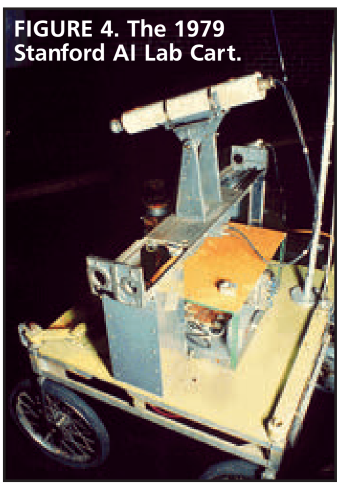
items as it was one of the products that changed the face of experimental/hobby robotics.

Figure 2 shows the later version of the Hero — the Hero 2000. Also, based also in Ann Arbor, MI — the home of the Robotic Industries Association — the NSRA had great initial enthusiasm but garnered little industry support. As did the RI/SME, the NSRA seemed to shrink to oblivion, though the parent group (RIA) is still active and presents many conferences and exhibits around the country.

Microsoft's Kinect Gives Robot Builders A New Goal

Robotics has changed in the past decades. Powerful and cheap computing power coupled with some amazing new sensors have given robot designers new directions and applications for service robots that were not only unavailable decades ago, but unthinkable. *Electronic Design's* April issue featured Willow Garage's PR-2 on the cover with the associated article "Robot Revolution" addressing how 'Cooperation Leads to Smarter Robots.' *Desktop Engineering's* July issue featured an article entitled 'Mobilizing Toward a Robotics Revolution' that delved into intelligent robot sensors, in particular, the Microsoft XBOX Kinect shown in **Figure 3**.

FIGURE 4. The 1979 Stanford AI Lab Cart.



This amazing device has taken the intelligent robotics field by storm the past couple of years and seems to have surpassed bipedal and self-balancing robots as the biggest subject of interest for advanced robot builders. Robot builders are now able to use this device to allow their robots to not only sense human presence, but to allow the sensed humans the ability to control the robot through hand and body motions without the need for a wired hand controller.

For years, artificial intelligence was limited to a handful of basic sensors to deliver a perception of the robot's world to its processor. Ultrasonic and IR sensors could reach out and detect obstacles that might affect the robot's path. Passive sensors could detect sound, light, and even understand a human's speech for control. Inventors attached web cams and other visual imaging devices to allow a robot to perceive the outside world. Stanford University's AI Lab had built the 'AI Lab Cart' shown in **Figure 4** and 'Shakey' in the '70s and '80s — robots that used crude vidicon camera vision systems to navigate. The results were amazing for the time but lacked usefulness compared with today's technology. Over a decade later, Sony's Aibo of the '90s was capable of recognizing human faces, and robot experimenters soon applied this technology to their creations.

Kinect Is An Ideal Intelligent Vision System For A Robot

It was not until Microsoft introduced the $149.95 Kinect peripheral for their Xbox-360 that robot builders with a limited budget could now create a machine that would make sense of human motion. On November 4th of last year, Microsoft released the Kinect, and Xbox gamers jumped on it like bees on a picnic watermelon. So did robot experimenters. Just as the Scarecrow and Tin Man in the *Wizard of Oz* felt they needed a brain or heart to finally be accepted as intelligent entities, robot builders wanted their robots to have a true, functional vision system. This group of experimenters saw this new product as a dream come true for their creations — at least as far as the vision part goes.

By March of this year, Microsoft had sold over 10 million of the devices

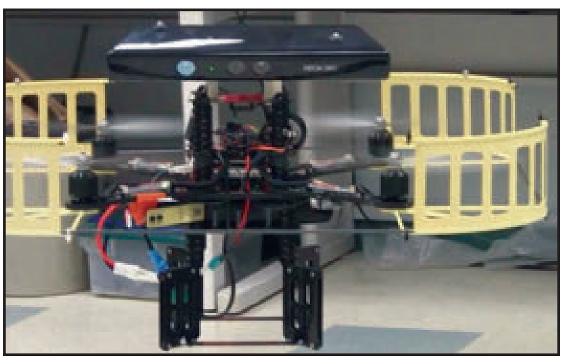


FIGURE 5. Kinect Quadrotor from IEEE *Spectrum*.

— far out-selling Apple's i-Phone and i-Pad launch sales. The device was a natural for hackers. When an ex-Microsoft employee offered $3,000 for the first person to hack the system, it was hacked within days of its launch. You-Tube has a slew of videos of Kinect-powered robots. The photo from the IEEE *Spectrum Magazine* in **Figure 5** shows a Kinect-controlled Quad Rotor AAV. You can see the four whirling rotors protected by the yellow 'fences.'

Xbox gamers saw the jewel that it was, but it was the robot experimenter community that dug into the amazing sensor and made it the center of their robotic creations. Universities around the world saw the Kinect as a replacement for previous complex systems that cost tens of thousands of dollars and did not work as well as the $150 Microsoft product. At first, Microsoft — as with most large companies — strenuously objected to the hacking competition but soon realized that making the device 'open source' would benefit everyone — especially them.

The Kinect enables control of the Xbox through "natural interaction" — a term trademarked by PrimeSense (Tel Aviv, Israel) — the company that developed Kinect's underlying optical sensing and recognition technology that translates body motion into control commands. It works by projecting an infrared laser pattern onto nearby people and objects. A dedicated IR sensor picks up on the laser to determine distance for each pixel, and that information is then mapped onto an image on a standard RGB camera. The Kinect sensor has an IR emitter, a depth camera coupled with a standard RGB camera, and a built-in array of four microphones that track your full-body movements and respond to your voice.

The Kinect project started out as Project Natal after they bought another Israeli startup company (3DV for $35 million in early 2009). 3DV was a developer of 3D real time depth detection digital cameras. PrimeSense and Microsoft examined technology from both companies and ended up using the PrimeSensor technology. **Figure 6** shows one of the earlier, pre-Kinect PrimeSense-WAVI-Xtion systems. Note the similarity to the later Kinect configuration.

Figure 3 shows the simple layout of the Kinect sensor system and the three lenses. When the projected IR pattern from the emitter seen as the far left 'lens' hits a 3D object, the lines are distorted and this distortion is read by the depth camera (the camera to the right). This camera analyzes the distorted IR patterns and builds a 3D map of the room with all the objects and people in it.

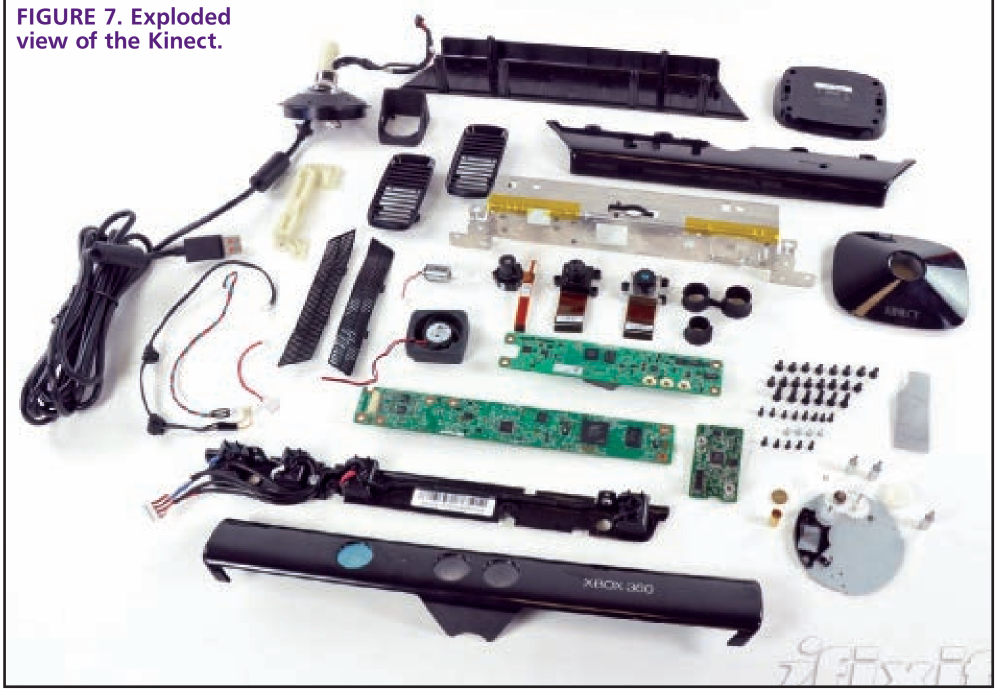
FIGURE 6. The PrimeSense WAVI-Xtion 3D sensor.

The center camera is the color camera that is used to gather details about people and objects in the viewing area. **Figure 7** shows an exploded, hacked view of the Kinect. From the bottom, you can see the four-microphone array, the three circuit boards, and the three cameras. I'm not going to elaborate on the aspects of Kinect as you can search the Internet and find dozens of sites explaining the device, as well as dozens of hacking sites and hundreds of robots using the Kinect.

Other Changes In Robotics Over The Years

There are many other technological breakthroughs that did not necessarily cause changes in the field of robots, but rather allowed these changes to occur. One might say that low cost GPS modules (such as the two Parallax GPS modules that cost $35 and $80 each) have allowed builders

FIGURE 7. Exploded view of the Kinect.



to make robots that can traverse outdoors and navigate fairly accurately. The Parallax 28146 module at $79.99 is shown in **Figure 8**, but there are many companies who offer similar modules for experimenters. Others have hacked inexpensive handheld GPS receivers for the same purpose, and hacking information is available on the Web. Needless to say, the accuracy is not what we might expect for a robot that operates indoors within a 10 foot square room, but GPS navigation is ideal for outside events such as the Robo-Magellan contests that have become quite popular across the country.

First envisioned by the Seattle Robotics Society, builders have constructed winning Robo-Magellan entrants for little more than a few hundred dollars in parts. Without the cheap CCD/CMOS laptop web cameras and GPS modules available to robot hobbyists today, robots such as these were unthinkable just 20 years ago outside of university and industry labs. Now, builders can toss in some next step ideas such as high power density LiPo and other battery designs, rare-earth PM and brushless DC motors, cheap microcontrollers, and a wealth of information available on the Internet. 3 axis gyroscopes and accelerometers make stable walking robots a possibility.

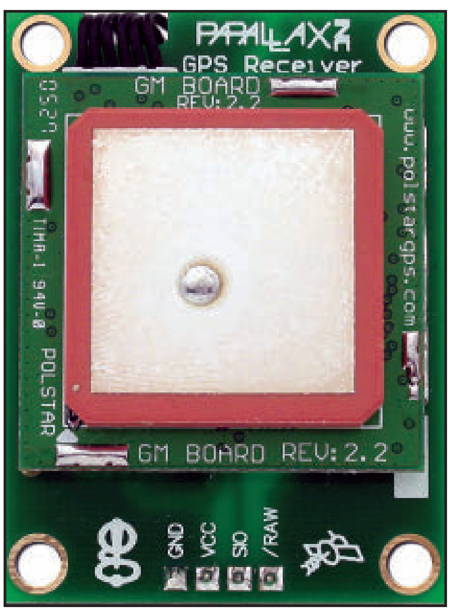


FIGURE 8. The Parallax GPS sensor receiver.

Closing Thoughts

This past June, President Obama announced the National Robotics Initiative — a program to develop next-generation robots for industrial purposes, healthcare, and other service robot applications. Robotics groups were enthusiastic about this news, but as various administrations in the past had proven with previous announcements about new technology, most will take a wait and see attitude. Will such an initiative prove to be a way for our industries to replace skilled technical and production people with brainless button-pushers? Are we, indeed, heading to another robotics revolution, as the non-technical media has touted for so long? Will these changes in sensor, computing, vision, and power systems technology create the changes in robotics that we are all striving towards? We'll just have to wait and see. **SV**

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

Advertiser Index

All Electronics Corp. 47, 81
Anaren 61
AndyMark 54, 81
AP Circuits 9
BaneBots 7
ComBots 19
Firgelli 17
HiTec 2, 81
Hobby Horse 54
Lynxmotion, Inc. 13
Maxbotix 81
Minds-i 80
PCB Pool 81, 82
Pololu Robotics & Electronics .3, 81
RobotShop, Inc 66, 81
Robot Power 9
Servo City/Robot Zone 83
Solarbotics/HVW 7
Sunstone Circuits ....Back Cover
superbrightleds.com 81
The Robot Marketplace 55
Vantec 55
Weirdstuff Warehouse 47